Technische Universität München

Fakulät für Informatik

Master's Thesis in Informatik

Konzepte zur Priorisierung von Alarmen in heterogenen IDS-Umgebungen

Richard Kaufhold, B.Sc.

Technische Universität München

Fakulät für Informatik

Master's Thesis in Informatik

Konzepte zur Priorisierung von Alarmen in heterogenen IDS-Umgebungen

Concepts for Alert Priorization in heterogeneous IDS environments

Autor	Richard Kaufhold, B.Sc.
Aufgabensteller	Prof. Dr.-Ing. Georg Carle
Betreuer	Nadine Herold, M.Sc. ; Dipl.-Inf. Stephan-A. Posselt
Datum	15. März 2015

Informatik VIII
Lehrstuhl für Netzarchitekturen und Netzdienste

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Garching b. München, 15. März 2015

Unterschrift

Zusammenfassung

Computernetzwerke sind permanent in Gefahr angegriffen zu werden. Deshalb wurden NIDS-Systeme (Network Intrusion Detection System) entwickelt, die Ungewöhnliches und Bedrohliches in Form von Alarmen melden. Die meisten Netzwerkkomponenten verfügen über eine Reihe von Abwehr- und Schutzmechanismen, sodass ein Angreifer mehrere, mehr oder weniger komplexe Teilangriffe benötigt, um sein Ziel zu erreichen. Dies führt zu einer grossen Menge an Alarmen in kürzester Zeit, was eine der grössten Schwachstellen dieser Systeme ist. Das Ziel dieser Arbeit ist, in solchen Hochlast-Situationen eine zeitnahe Abarbeitung der wichtigen Alarme durch eine intelligente Priorisierung zu gewährleisten. Dafür wurde der Ansatz gewählt, Informationen aus verschiedenen Datenquellen, die in Verbindung mit den Sicherheitsalarmen stehen, zu kombinieren. Es wurden Informationsquellen wie z.B. Netzwerktopologie, Dienststruktur oder Schwachstellendatenbanken identifiziert, deren Kombinationsmöglichkeiten ausgearbeitet und Wege aufgezeigt, wie diese automatisiert erfasst werden können. Um aus den verschiedenen Informationen zu einem Alarm dessen Prioritäts-Wert ermitteln zu können, wurde eine Berechnungsvorschrift entwickelt, die sie vereint. Die Evaluierung des Ansatzes wurde unter anderem mittels eines Prototypen durchgeführt und führte zu einem positiven Ergebnis. Der Ansatz erfüllte alle Kriterien an eine erfolgreiche Priorisierung, die nach einer detaillierten Analyse bestehender Ansätze aufgestellt wurden, und zeigte mehrere Vorteile auf, die zu einer positiven Bewertung der wichtigsten Eigenschaften führten.

Abstract

Computer networks are permanently in danger of being attacked. Consequently, Network Intrusion Detection Systems (NIDS) have been developed for the purpose of detecting abnormal behaviour or malicious activity and signaling them in the form of alarms. Most of the network components are provided with defence and protection mechanisms so that an attacker needs several more or less sophisticated and complex sub-attacks in order to achieve his goal. This results in a large number of alarms being issued within a very short time and this amounts to one of the most critical vulnerabilities of such network systems. The purpose of this work has been to assess the possibility of ensuring prompt processing of important alerts during such high-load situations by way of intelligent prioritization. The approach chosen for achieving this purpose has been to combine information from various data sources pertinent to the security alerts. This was achieved by identifying relevant information sources such as network topology, service structure or vulnerability databases, devising possibilities of combining the information provided by them and pointing out ways of collecting such information in an automated manner. For the purpose of determining the priority value of an alarm from the various information relating to it, a calculation rule for combining such information was developed. The evaluation of this approach was carried out by means of, among other, a prototype, and it yielded a positive result. The approach fulfilled all criteria for successful prioritization that had been established after a detailed analysis of already known approaches, and it featured several advantages which led to a positive evaluation result in respect of the most important requirements.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Unbekannte konnten sich bei ihrem Angriff auf die Handelsplattform eBay Zugang zu 145 Millionen Datensätzen mit Name, E-Mail-Adresse, Postadresse, Telefonnummer, Geburtsdatum und dem verschlüsselten Passwort der Kunden verschaffen [8]. Solche Vorfälle zeigen uns die Wichtigkeit einer fortlaufenden Kontrolle der Firmennetzwerke und die Notwendigkeit einer schnellen Reaktion auf Einbrüche. Um dies zu erreichen, kann der Netzwerkverkehr mit Hilfe von Network-Intrusion-Detection-Systemen (NIDS) analysiert werden, die sich durch die Art des Monitorings unterscheiden: Signatur-basierte Systeme sind solche, die nach bekannten Angriffsmustern suchen, Anomalie-basierte Systeme werden durch ungewöhnlichen Datenverkehr aktiviert.

Passiert etwas Ungewöhnliches, das durch ein NIDS als bedrohlich eingestuft wird, so übermittelt dieses eine Alarmmeldung an die Systeme, die sich für die Benachrichtigung über Störfälle registriert haben. Die meisten Netzwerkkomponenten verfügen über eine Reihe von Abwehr- und Schutzmechanismen, sodass ein Angreifer mehrere mehr oder weniger komplexe Teilangriffe benötigt, um sein Ziel zu erreichen. Das führt zu einer grossen Menge an Alarmen in kürzester Zeit, die ein Intrusion-Detection-System (IDS) produziert. Dies stellt eine der größten Schwachstellen dieser Systeme dar. Netzwerkadministratoren, die mit der Absicherung und der Kontrolle des Netzwerks betraut sind, können durch die Vielzahl der Alarmmeldungen schnell überfordert werden. Dadurch steigt die Wahrscheinlichkeit, dass wichtige Ereignisse, welche einen Angriff auf bzw. einen Einbruch in das gesicherte Netzwerk aufdecken, zu spät erkannt oder übersehen werden. Einer der Wege, diesem Problem zu begegnen, ist die Sicherheitsalarme zu priorisieren, sie entsprechend ihrer Wichtigkeit in eine Rangfolge zu bringen.

Hauptziel dieser Arbeit ist, aufzuzeigen, wie in Hochlast-Situationen bei einem Angriff auf ein Netzwerk eine zeitnahe Abarbeitung der von Intrusion-Detection-Systemen gesendeten wichtigen Alarme durch eine intelligente Priorisierung gewährleistet werden kann.

Um eine derartige Priorisierung zu erzeugen, wird der Ansatz gewählt, Informationen

aus verschiedenen Datenquellen, die in Verbindung mit den Alarmen stehen, zu kombinieren. Es wurden bereits mehrere unterschiedliche Ansätze zur Priorisierung von Alarmen entwickelt, die es zu analysieren gilt. Vor allem sollten solche ausgesucht und genauer betrachtet werden, die unterschiedliche Informationsquellen verwenden. Ein weiteres Ziel dieser Arbeit ist es, Daten- bzw. Informationsquellen zu identifizieren und bezüglich ihrer Einsatzmöglichkeiten für die Priorisierung der Alarme zu evaluieren. Des Weiteren sollen deren Vor- und Nachteile bei der Ausarbeitung des Ansatzes betrachtet werden.

1.1 Aufgabenstellung

Um die Aufgabenstellung zu konkretisieren und die Arbeit an der Entwicklung des Priorisierungsansatzes, des dazugehörigen Prototypen und deren Evaluierung zu strukturieren, werden folgende wissenschaftliche Fragen formuliert:

F.1 Welche Informationsquellen sind für die Priorisierung von Sicherheitsalarmen relevant?

F.2 Wie können diese Informationsquellen miteinander kombiniert werden, um sie für die Priorisierung zu verwenden?

F.3 Inwieweit kann der Prozess der Erfassung und Verarbeitung der Informationen aus diesen Quellen automatisiert werden?

Diese Fragen gilt es im Rahmen der vorliegenden Arbeit zu beantworten.

1.2 Kapitelübersicht

Zu Beginn der Arbeit werden bereits vorhandene Ansätze für die Priorisierung von Sicherheitsalarmen analysiert. Einige ausgewählte Ergebnisse dieser Analyse werden im Kapitel 2 präsentiert. Dabei wird der jeweilige Ansatz dargestellt und seine Vor- und Nachteile werden kurz diskutiert. Eine abschließende Diskussion und ein Vergleich der Ansätze anhand bestimmter wichtiger Merkmale wie Skalierbarkeit, Automatisierbarkeit oder Flexibilität, zeigen die jeweiligen Stärken, die beachtet, und Schwächen, die vermieden werden sollten. Als Ergebnis der Analyse werden Anforderungen spezifiziert, die eine erfolgreiche Priorisierung erfüllen sollte.

Um auf die gestellten wissenschaftlichen Fragen antworten zu können, werden im Kapitel 3 die Datenquellen dargestellt, welche die für die Priorisierung relevanten Informationen liefern. Bei der Beschreibung wird darauf eingegangen, wie genau die Informationen zur Priorisierung der Alarme beitragen können und welche Möglichkeiten für die automatische Erfassung zur Verfügung stehen. Die abschließende Diskussion

der möglichen Kombinationen der Informationsquellen umfasst dann die Antwort auf die letzte der drei oben formulierten wissenschaftlichen Fragen.

Basierend auf den Erkenntnissen aus den beiden Analysen wurde ein Ansatz ausgearbeitet, dessen mathematische Repräsentation, Grundlagen und fachliche Hintergründe im Kapitel 4 detailliert dargestellt werden. Dabei wird begleitend ein Beispiel erläutert, um einzelne Schritte in den Berechnungen sowie die Berechnungsvorschriften besser erklären zu können.

Im Kapitel 5 wird nach der Erläuterung der theoretischen Grundlagen erklärt, wie das beschriebene Priorisierungsystem in Softwaremodule umgesetzt werden kann. Hierbei gilt es, einen besonderes Augenmerk auf die notwendigen Komponenten, das Datenmodell und die ausgeführten Prozesse zu richten.

Das Ergebnis dieser Umsetzung wird dann im Kapitel 6 beschrieben. Bei dieser Gelegenheit wird neben den verwendeten Techniken auch die tatsächliche Struktur des Prototypen genauer beleuchtet werden.

Um beurteilen zu können, ob die Ziele dieser Arbeit erreicht und die im Kapitel 2 aufgestellten Anforderungen durch den entwickelten Ansatz erfüllt sind, wird im Kapitel 7 eine Bewertung sowohl der prototypischen Implementierung als auch der mathematischen Gleichungen durchgeführt.

Abschließend werden in Kapitel 8 einzelne Schritte der durchgeführten Arbeit von der Aufstellung der Ziele bis zur Evaluierung in einer Zusammenfassung dargestellt und ein Ausblick auf weitere Entwicklungsmöglichkeiten präsentiert.

Kapitel 2

Verwandte Ansätze

In diesem Kapitel werden ausgewählte bestehende Ansätze zur Priorisierung der Sicherheitsalarme vorgestellt, um eine Übersicht über verschiedene Techniken und Möglichkeiten zu gewinnen. Es werden je Ansatz zuerst die allgemeine Vorgehensweise und die eingesetzte Technik vorgestellt. Anschließend werden die Vor- und Nachteile des Ansatzes kurz diskutiert. Nachdem alle Ansätze erläutert wurden, werden am Ende des Kapitels eine zusammenfassende Übersicht der Stärken und Schwächen präsentiert und die gravierendsten Nachteile aufgelistet, die es bei der Entwicklung des eigenen Ansatzes zu vermeiden gilt.

2.1 Risk Index Model (RIM)

RIM wurde im Jahr 2012 im Forschungsartikel [1] als ein möglicher Ansatz zur Bewertung und Priorisierung von Alarmen mit Hilfe des Analytischen Hierarchieprozesses (AHP) vorgestellt.

Der AHP ist ein Verfahren zur Unterstützung der Entscheidungsfindung bei komplexen Problemen. Er definiert systematisch die Schritte, die dazu nötig sind, einen hierarchischen Lösungsprozess für ein Problem festzulegen. Die Aufstellung des Lösungsprozesses mittels AHP gliedert sich nach [9] in vier Schritte:

1. Im ersten Schritt wird die Problemstellung und das gesuchte Ergebnis festgelegt.
2. Danach erfolgt die Aufstellung der Entscheidungshierarchie mit dem Ziel, d.h. dem gesuchten Ergebnis, auf der obersten Stufe. Die festgelegte Problemstellung der obersten Stufe wird in der nächst niedrigeren Stufe in Teilprobleme zerlegt. Diese werden dann in weiteren Hierarchiestufen wieder untergliedert, bis die benötigten Ergebnisse der Teilprobleme einfach ermittelbar sind.

3. Im nächsten Schritt werden paarweise Vergleichsmatrizen (Entscheidungsmatrizen) aufgestellt, die aufzeigen, wie sich die Ergebnisse einer Stufe, Alternativen genannt, auf ein Ergebnis der nächsthöheren Stufe abbilden lassen.

4. Die im vorangegangenen Schritt erzeugten Matrizen werden dazu verwendet, um Gewichte für die Abbildung der Lösungen der untergeordneten Teilprobleme auf die nächsthöhere Ebene zu berechnen. Dieser Prozess wird für jede Ebene der Entscheidungshierarchie wiederholt.

Bei den aufgestellten Entscheidungsmatrizen ist zu beachten, dass die Diagonalelemente den Wert 1 enthalten und die Elemente der Matrix oberhalb der Diagonalen zu den jeweiligen Elementen unterhalb der Diagonalen das multiplikative Inverse enthalten. In [9] wird außerdem ein Wertebereich der einzelnen Matrixelemente von $[1, 9]$ empfohlen wobei 1 bedeutet, dass die Alternative der Zeile gleich wichtig ist wie die der Spalte. Ein Wert von 9 bedeutet, dass die Alternative der Zeile ungleich wichtiger ist als die der Spalte.

In [1] wird ein dreistufiger Hierarchieprozess für die Priorisierung von Alarmen vorgeschlagen. Dieser ist in der Abbildung 2.1 schematisch dargestellt.

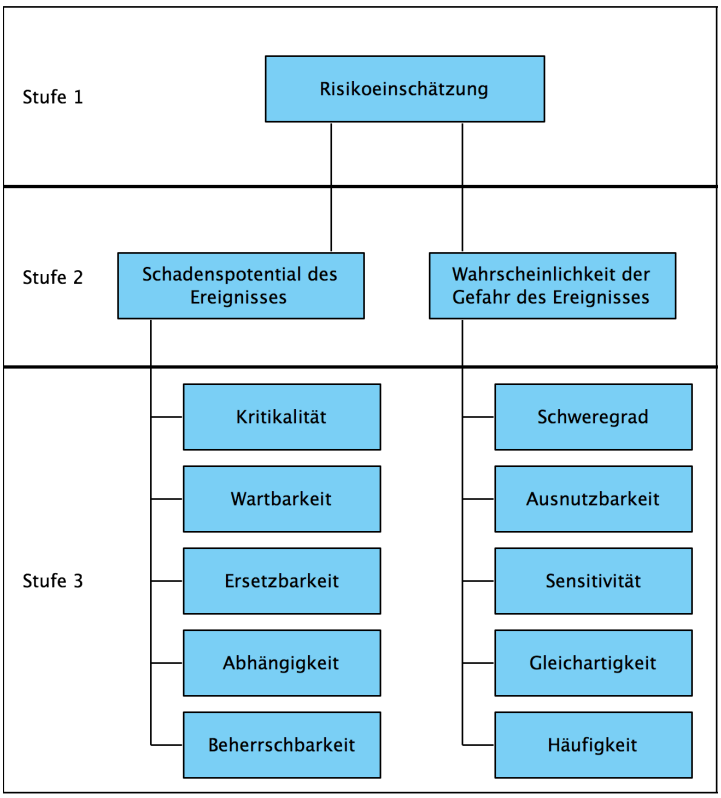


Abbildung 2.1: RIM AHP nach [1]

Diesem Prozess folgend werden nach Eingang einer IDS-Meldung zur Ermittlung der

Priorität die folgenden drei Hierarchiestufen durchlaufen:

- 3. Stufe: Zu Beginn werden die Werte der untersten Hierarchiestufe, in [1] Indikatoren genannt, für den gegebenen Alarm ermittelt. Thematisch werden die Indikatoren in die Gruppen Schadenspotential des Ereignisses und Wahrscheinlichkeit des Eintritts des Ereignisses aufgeteilt.
- 2. Stufe: Die zuvor ermittelten Indikatoren werden innerhalb ihrer Gruppe mittels der Gewichte aus den Entscheidungsmatrizen bewertet und zu einem Wert pro Gruppe kombiniert.
- 1. Stufe: Die in Stufe 2 errechneten Kennzahlen werden wiederum gewichtet und zu einer Gesamtpriorität vereint. [10]

Die erste Gruppe (Schadenspotential des Ereignisses) der Indikatorwerte aus Stufe 3 fasst alle Indikatoren zusammen, die zu der Aussage beitragen, wie stark die gemeldete Störung das System negativ beeinflusst. Es wurden die folgenden Fünf ausgewählt:

- Kritikalität (criticality) beschreibt, wie wichtig die angegriffene Komponente des Netzwerkes im Hinblick auf Vertraulichkeit, Integrität und Erreichbarkeit ist.
- Wartbarkeit (maintainability) gibt den monetären Wert der Instandhaltung der Komponente an.
- Ersetzbarkeit (replaceability) wird in [1] als ein Maß dafür definiert, wie einfach die Komponente unter Beachtung des Zeit- und Kostenfaktors ersetzt werden kann.
- Abhängigkeit (dependability) zeigt auf, wie stark andere Komponenten im Netzwerk von dem angegriffenen System abhängen.
- Beherrschbarkeit (control) ist ein Maß dafür, wie gut die auf der Komponente getroffenen Maßnahmen zur Verringerung der Auswirkungen von Angriffen und Sicherheitslücken beitragen. [1]

In der zweiten Gruppe (Wahrscheinlichkeit des Eintritts des Ereignisses) werden solche Indikatoren betrachtet, die das Ausmaß der Gefährdung und der Sicherheitslücke beziffern können. Diese sind im RIM-Ansatz:

- Der Schweregrad (severity) hängt von dem potentiellen Vorfall ab. Eine Abschätzung kann beispielsweise anhand der Sicherheitslücke getroffen werden.
- Ausnutzbarkeit (exploitability) beschreibt den Status der in der Komponente vorhandenen Sicherheitslücke und wie gut diese zu einem bestimmten Zeitpunkt für einen Angriff verwendet werden kann.
- Sensitivität (sensitivity) stellt die Beurteilung der Priorität der Meldung durch den erzeugenden Sensor dar.

- Gleichartigkeit (similarity) gibt den Ähnlichkeitsgrad des Alarms zu vorhergehenden Ereignissen an.
- Durch die Häufigkeit (frequency) wird beschrieben, wie oft ein ähnlicher Angriff in einem vordefinierten Zeitraum stattfand.

Durch die Verwendung einer Vielzahl von Indikatoren kann für jeden Alarm eine Priorität ermittelt werden [1]. Die Evaluierung des Ansatzes in [10] hat gezeigt, dass „100 % der Vorkommnisse von RIM bewertet werden konnten“, auch wenn Werte von einigen Indikatoren fehlten.

Der Ansatz, die Alarme mit RIM zu priorisieren, kann auf verschiedene Netzwerke übertragen werden. Die Größe des Netzwerkes schlägt sich dabei nicht in der Komplexität oder im Ressourcenverbrauch der Berechnungen nieder.

Die Werte der Prioritäten können bei RIM sehr feingranular sein, so dass ein besseres Ranking der Alarme untereinander möglich ist. Die Wahrscheinlichkeit, dass zwei Alarme dieselbe Priorität erhalten, ist somit geringer.

Die für RIM aufgestellten Entscheidungsmatrizen können mathematisch verifiziert werden, wodurch die Dominanz eines einzelnen Indikators vermieden werden kann.

Trotz all dieser Vorteile sind zu diesem Verfahren einige Nachteile anzumerken.

Die Erstellung der Entscheidungsmatrizen muss von Experten manuell durchgeführt und bei Netzwerkänderungen angepasst werden. Durch die mathematische Verifikation der Entscheidungsmatrizen kann zwar die Dominanz einzelner Indikatoren ausgeschlossen werden, dies liefert jedoch keine Garantie, dass die resultierende Gewichtung zu dem gegebenen Netzwerk passt.

Hinzu kommt die Notwendigkeit einer manuellen Eingabe vieler Indikatorwerte, da diese nicht automatisiert ermittelbar sind.

Zudem müssen die Indikatoren ausgiebig getestet werden, um die Wahrscheinlichkeit einer fehlerhaften Priorisierung zu verringern.

In [10] wurde die Anzahl der Indikatoren für jede Gruppe auf fünf reduziert, um die Komplexität der Berechnung zu verringern. Die Erweiterung um mehr Indikatoren würde die Präzision der Bewertungen steigern (durch die Einbeziehung von mehr Informationsquellen), jedoch auch die Komplexität der Berechnung der Matrizen erhöhen.

2.2 FuzMet

Ein weiterer Ansatz, wie man mit der großen Menge von IDS-Alarmen umgehen und ihre Prioritäten richtig bewerten kann, wird im Forschungsartikel zu FuzMet [11] vorgestellt.

Die Prioritätsberechnung von FuzMet basiert auf Fuzzy-Logic. Die Fuzzy-Logic stellt eine Methode zur Verbindung von scharfen und unscharfen Mengen (Fuzzy-Mengen) dar. In Abbildung 2.2 sind die Schritte zur Anwendung der Fuzzy-Logic schematisch dargestellt. Die Ein- bzw. Ausgabeparameter der Berechnung sind hierbei Werte aus scharfen Mengen.

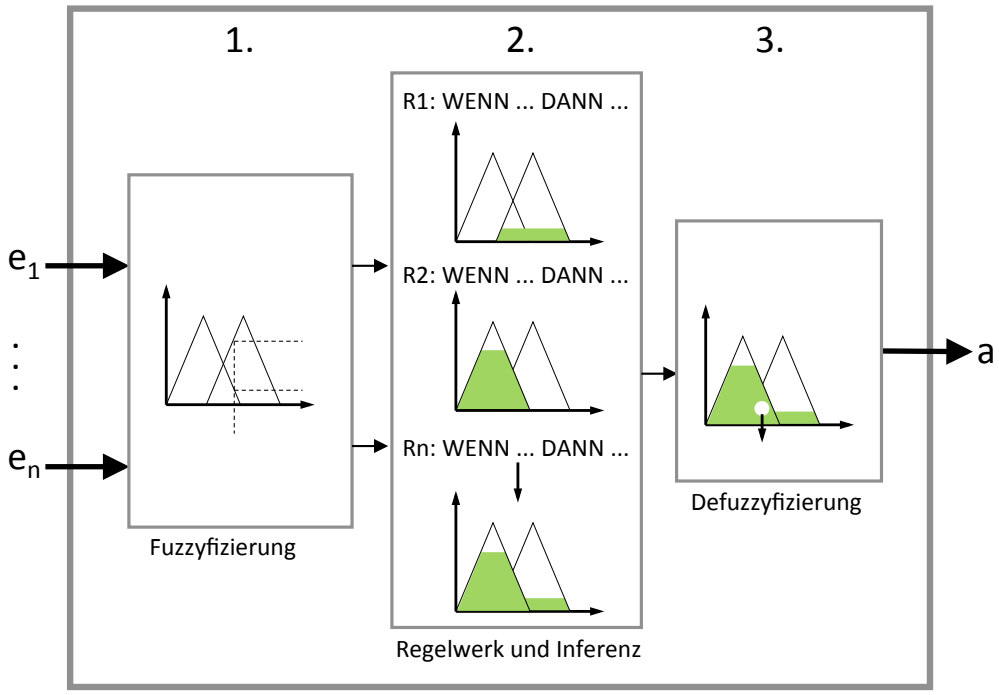


Abbildung 2.2: Fuzzy-Logic-Ablauf nach [2]

Im 1. Schritt werden die Eingabeparameter mittels Zugehörigkeitsfunktionen auf so genannte Fuzzy-Mengen abgebildet. Das bedeutet, es wird für den scharfen Wert die prozentuale Zugehörigkeit zu allen Elementen der Fuzzy-Mengen berechnet.

Im 2. Schritt werden die Werte der Abbildungen verwendet, um vorher definierte Regeln zu aktivieren, wodurch Ausgabewerte generiert werden. Die Regeln sind hierbei basierend auf den unscharfen Mengen definiert und benutzen deren Logikkalkül [12] für die Berechnung der Ausgabewerte. Des Weiteren wird in diesem Schritt auch festgelegt, wie die durch mehrere Regeln generierten Ausgabewerte zusammengeführt werden.

Im 3. Schritt werden die so erzeugten Werte aus den Ausgabe-Fuzzy-Mengen mittels einer für die konkrete Menge passenden Methode defussifiziert, d.h. wieder auf einen scharfen Wert abgebildet.

Dem System der Fuzzy-Mengen folgend, ist auch bei FuzMet der erste Schritt zur Prio-

risierung eines Alarms die Erhebung der Eingabewerte, hier Metriken genannt. In [11] wurden die folgenden sieben Metriken ausgewählt:

- Anwendbarkeit des Angriffes auf das Netzwerk (applicability) bedeutet eine Prüfung anhand der ausgenutzten Schwachstelle, ob das mögliche Ziel des Angriffes im Netzwerk vorhanden, nicht ausreichend geschützt und zum Zeitpunkt des Angriffes erreichbar ist.
- Wichtigkeit des Ziels (importance of victim) gibt die Bewertung der Wichtigkeit des als mögliches Ziel für den Angriff spezifizierten Hosts für das Netzwerk an. Die Wichtigkeit des Hosts wird hierbei gemäß den auf dem Host installierten Anwendungen, der von ihm angebotenen Dienste und der Benutzerkonten, die er verwaltet, bewertet.
- Vertrauen in die Meldung des Sensors (sensor status) beinhaltet die Wahrscheinlichkeiten richtig-positiver, falsch-positiver, richtig-negativer und falsch-negativer Messungen, bezogen auf das IDS, das den Alarm gemeldet hatte.
- Wichtigkeit der Platzierung des Sensors (sensor placement) gemessen an einer Abschätzung der Wichtigkeit des Subnetzwerkes, das hinter dem IDS-Sensor liegt.
- Schweregrad (severity) gibt, abgeleitet aus Angaben unterschiedlicher Quellen, die Bedeutung des Alarmes im Hinblick auf die ausgenutzte Schwachstelle an. Die Quellen, die hierbei zum Einsatz kommen, sind beispielsweise MITRE CVE, NIST, Secunia oder OSVDB.
- Angreifbarkeit bzw. Stabilität des Dienstes (service vulnerability) stellt die kombinierte Bewertung des Schweregrades der bereits behobenen und momentan bekannten Schwachstellen des angegriffenen Dienstes unter Beachtung der Anzahl der Monate seit der Veröffentlichung des Dienstes dar.
- Zusammenhänge zwischen den Alarmen (alert relationship), gemessen anhand der Ähnlichkeit der Attribute, werden verwendet, um mehrstufige Angriffe höher zu bewerten. [11]

Die so erhobenen Werte werden pro Metrik anhand von Zugehörigkeitsfunktionen, die auf der Normalverteilung basieren, auf Fuzzy-Mengen mit den Werten „Low“, „Med“, und „High“ abgebildet. Die Ergebnisse dieser Abbildung fließen dann in den Schritt der Auswertung der Fuzzy-Regeln ein. Hierbei sind die Regeln so gestaltet, dass sie die Eingabe-Fuzzy-Mengen auf die Prioritäts-Fuzzy-Menge („alert score“) abbilden. Die aus den Regelauswertungen resultierenden Ergebnisse werden mit Hilfe der Mittelwertbildung zu einem Ergebnis kombiniert. Um einen scharfen Wert für die Priorität zu erhalten, wird der im letzten Schritt kombinierte Wert durch die Berechnung des Massenschwerpunktes defuzzyfiziert. In einem nachgelagerten Schritt können die niedrig priorisierten Alarme mit einer höheren Priorität versehen werden, falls ein starker Zusammenhang zu einem hoch priorisierten Alarm festgestellt wird (alert rescoring).

Dieses Rescoring kann als eine positive Eigenschaft des FuzMet-Ansatzes gesehen werden. Zusammen mit der Metrik für die Zusammenhänge zwischen den Alarmen werden so mehrstufige Angriffe besser priorisiert.

Die in der Fuzzy-Logic verwendeten Berechnungen können sehr schnell ausgeführt und zu einem großen Teil auch parallelisiert werden, wodurch das System bei ausreichender Rechenleistung auch mit sehr vielen Alarmen in einem kurzen Zeitraum umgehen kann.

Da bei diesem Verfahren die Berechnungen während der Ermittlung der Prioritäten von der Größe des Netzwerks unabhängig sind, wirken sich die Veränderungen in der Größe des Netzwerkes nicht auf die Komplexität oder Ausführungsgeschwindigkeit der Berechnungen aus.

Als nachteilig angesehen werden kann allerdings, dass die Erfassung der Werte von einigen der Metriken sehr rechen- bzw. zeitintensiv werden kann. Ein Beispiel ist die Metrik „alert relationship“, bei der die Attribute des Alarms mit den Attributen einer großen Menge anderer Alarme verglichen werden.

Außerdem benötigt die Spezifikation der Fuzzy-Regeln das Expertenwissen über die Fuzzy-Logic und das zu schützende Netzwerk.

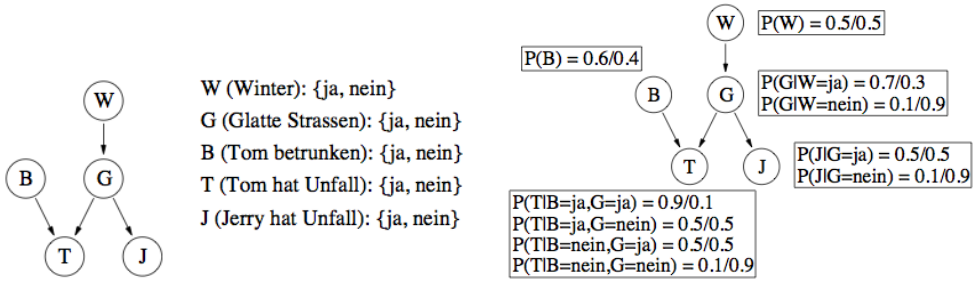
In Steuerungssystemen, die Fuzzy-Logic einsetzen, ist meist eine Rückkopplung der Ausgabeparameter der Fuzzy-Logic zu den Eingabeparametern innerhalb des zu steuernden Prozesses gegeben. Diese Rückkopplung ermöglicht es, die Ausgabewerte der Fuzzy-Logic auf ihre Korrektheit bzw. Stabilität hin zu bewerten, was bei FuzMet leider nicht der Fall ist.

2.3 M-Correlator

Ein weiterer Ansatz, der wie FuzMet versucht, möglichst viele verschiedene Faktoren bei der Priorisierung der Alarme zu beachten, wird in dem Patentantrag [13] und dem wissenschaftlichen Artikel [4] dargestellt. Der Artikel beschreibt das Softwaresystem M-Correlator als prototypische Umsetzung für die Minimierung der von Analysten zu prüfenden Alarme. Eine der verwendeten Methoden ist unter anderem die Priorisierung mittels Bayes-Netzen.

Bayes-Netze sind eine Struktur zur effizienten und „kompakten Speicherung und Verarbeitung unsicheren Wissens“ [3]. Diese Struktur stellt eine Erweiterung der kausalen Netze dar, die ihrerseits gerichtete azyklische Grafen (DAG’s) sind, mit Zufallsvariablen als Knoten und bedingten Abhängigkeiten zwischen ihnen als Kanten. Die Knoten besitzen dabei eine Menge von Zuständen und sind immer in genau einem davon, in welchem genau kann jedoch unbekannt sein. Wenn die Zustände eines Knoten direkte Ursache der Zustände eines anderen Knotens sein können, dann werden die Knoten durch eine gerichtete Kante verbunden. In Bayes-Netzen wird jeder Zufallsvariablen,

die eingehende Kanten hat, zusätzlich eine bedingte Wahrscheinlichkeitsverteilung im Form einer Wahrscheinlichkeitstabelle zugeordnet. Die Wahrscheinlichkeitstabellen der Knoten ohne eingehende Kanten, also die ohne Vorgänger-Knoten, beinhalten die absoluten Wahrscheinlichkeiten. Die Abbildung 2.3(b) zeigt ein einfaches Beispiel eines Bayes-Netzes, das aus dem kausalen Netz 2.3(a) abgeleitet wurde.



(a) Kausales Netz (b) Bayes-Netz

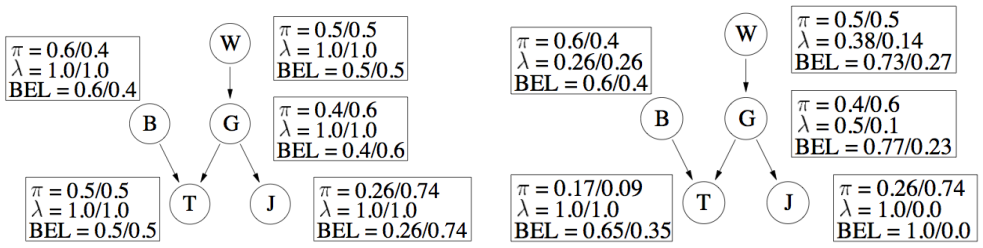
Abbildung 2.3: Beispiel Bayes-Netz [3]

Die Veränderungen im Netz werden mittels dem Verschicken von sogenannten λ-Nachrichten von Kind- zu Vorgänger-Knoten und π-Nachrichten von Vorgänger- zu Kind-Knoten propagiert. Diese Propagierung der Veränderung der Wahrscheinlichkeiten durch Nachrichtenversendung basiert auf der Definition, dass für jeden Zustand drei Werte gespeichert werden: „absolute Wahrscheinlichkeit des Zustandes“ als *BEL*, „Stärke der Unterstützung des Zustandes durch Vorgänger- bzw. Nachfolgervariable“ als π und λ [3]. Diese werden vor der Propagierung initialisiert. Alle λ werden mit 1 initialisiert. Die Werte für *BEL* und π werden nach folgenden Formeln errechnet, wobei zu beachten ist, dass *BEL* Werte eine Wahrscheinlichkeitsverteilung bilden und somit in der Summe 1 ergeben, was durch den Faktor β erreicht wird:

$$BEL(A = a_i) = \alpha\pi(A = a_i)\lambda(A = a_i) \tag{2.1}$$

$$\pi(G = ja) = P(G = ja|W = ja)BEL(W = ja) + P(G = ja|W = nein)BEL(W = nein) \tag{2.2}$$

Die Abbildung 2.4 stellt beispielhaft die Propagierung der Information „Jerry hat einen Autounfall“ dar. Hierbei ist in 2.4(a) das initialisierte Bayes-Netz dargestellt und in 2.4(b) der Zustand nach der Propagierung. Die Propagierung begann somit in der Zufallsvariable J, deren *BEL*-Wert sich nach dem Unfall auf 100 % bzw. 1 veränderte. Von J wurde dann eine λ-Nachricht an G geschickt und von G an W. Die Wahrscheinlichkeiten, dass die Straßen glatt sind und es Winter ist, sind somit auf jeweils 77 % und 73 % gestiegen. Des Weiteren wurden auch eine π-Nachricht von der Variable G an die Variable T und



(a) Initialisiertes Bayes-Netz (b) Bayes-Netz nach Propagierung

Abbildung 2.4: Beispiel einer Bayes-Netz Propagierung [3]

eine λ-Nachricht von der Variable T an die Variable B geschickt, wobei die letztere keine Veränderungen bewirkte. Die Wahrscheinlichkeit, dass Tom einen Unfall hat, ist nach der Propagierung auf 65 % gestiegen.

Zur Priorisierung von Alarmen wird in [4] das in Abb. 2.5 dargestellte Bayes-Netz vorgeschlagen. Der oberste Knoten stellt die Zufallsvariable Incident Rank (Priorität des Alarmes) mit der Menge der Zustände „low“, „medium“, „high“ dar. Es wird des Weiteren eine Funktion definiert, die diese Zustände auf die Prioritätswerte zwischen 0 und 255 abbildet. Der Wurzel-Knoten ist der Vorgänger der Knoten Outcome (Ergebnis), Priority (Priorität) und Relevance (Relevanz). Der Knoten Outcome beinhaltet die Aussage, ob der Angriff erfolgreich war oder nicht, was aus den Attributen des Alarmes abgeleitet wird. Die Zwischenvariable Priority setzt sich zusammen aus dem Knoten Attack Code Interest, mit einer von Experten festgelegten Wichtigkeit des vorliegenden Angriffstypen, und einer weiteren Zwischenvariablen Asset, die die Werte der Zufallsvariablen wichtiger Vermögenswerte zusammenfasst. Die Zwischenvariable Relevance setzt sich aus Zufallsvariablen zusammen, welche die Relevanz des Alarms für die angegriffene Komponente angeben.

Wird ein Alarm registriert, werden dessen Attribute verwendet, um die Wahrscheinlichkeiten der Blatt-Knoten zu bestimmen. Diese werden dann bis zum Wurzel-Knoten Incident Rank propagiert und beeinflussen die Wahrscheinlichkeiten von dessen Zuständen.

Ein großer Vorteil, der durch den Einsatz der Bayes-Netze entsteht, ist, dass das Netzwerk aus den eingehenden Alarmen und Eingaben von Experten lernen kann. Die Trainierung des Netzwerkes kann dabei auch während des Betriebs des Systems erfolgen.

Durch die vorgeschlagene Struktur des Bayes-Netzes ist es möglich, alle Informationen über den Alarm abzubilden und auch mit fehlenden Informationen umzugehen.

Die Art, wie das Netzwerk gestaltet ist, kann dazu genutzt werden, um weitere Informationsquellen einzubinden. Der M-Correlator kann an verschiedene Netzwerke angepasst werden.

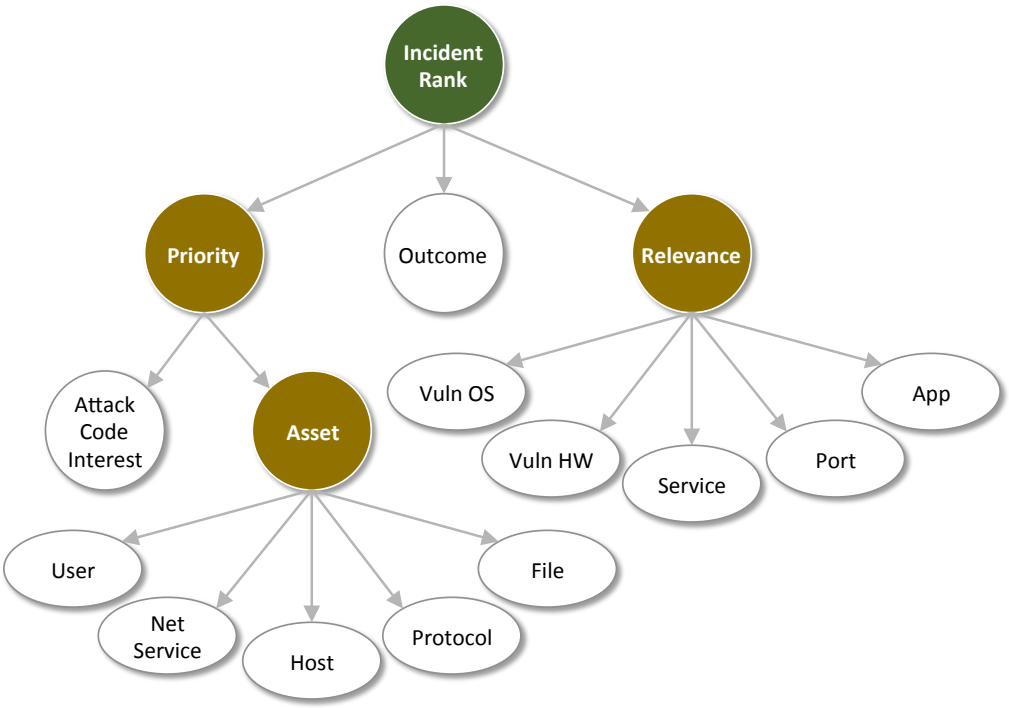


Abbildung 2.5: Bayes-Priorisierungsnetz nach [4]

Da der Aufwand für die Berechnung der Priorität nur von der Größe des Bayes-Netzwerks abhängt, kann das System auch auf große Rechnernetze angewendet werden.

Leider kann das Bayes-Netz erst nach der initialen Spezifikation der Wahrscheinlichkeits-Tabellen und einer Trainingsphase effektiv verwendet werden. Deshalb ist es nicht voll automatisierbar.

Nachteilig ist außerdem, dass dieser Trainingsprozess zeit- und ressourcenaufwändig ist.

Bei größeren Veränderungen des Netzwerks kann es zudem notwendig sein, die Wahrscheinlichkeits-Tabellen an die neuen Gegebenheiten anzupassen.

2.4 Common Vulnerability Scoring System (CVSS)

Das Common Vulnerability Scoring System (CVSS) ist ein Open-Source-Framework zur Spezifikation der Eigenschaften und Auswirkungen von IT-Sicherheitslücken. Die folgende Beschreibung bezieht sich auf CVSS V2, da sich V3 derzeit noch im Entwurf befindet. CVSS ist kein System zur Priorisierung von Alarmen, wie alle anderen Ansätze dieses Kapitels, sondern ein Ansatz zur Priorisierung von Sicherheitslücken. Dafür

werden Metriken definiert, die die Eigenschaften der Sicherheitslücke beschreiben. Diese sind in die drei Gruppen Base, Temporal, Environmental Metriken unterteilt.

Die Base Metriken vereinen die Charakteristika der Sicherheitslücken, die über die Zeit konstant und unabhängig von möglichen Umgebungen sind. Als solche werden der Zugriffsvektor (Access Vector, AV), die Komplexität des Zugriffs (Access Complexity, AC) und Authentifizierung (Authentication, Au) identifiziert, die angeben, welche Bedingungen erfüllt sein müssen, um den Zugriff auf die Sicherheitslücke zu erlangen. Des Weiteren werden drei Metriken definiert, die den Grad des Verlustes der Integrität (Integrity, I), der Vertraulichkeit (Confidentiality, C), sowie der Verfügbarkeit (Availability, A) durch die Verwendung der Sicherheitslücke angeben. Die Angabe der Werte dieser Gruppe ist verpflichtend und wird vom Herausgeber der Bewertung der Sicherheitslücke durchgeführt.

Die Temporal Metriken erfassen Eigenschaften, die sich über die Zeit des Bestehens der Sicherheitslücke ändern können. Die drei Metriken geben an, wie einfach die Sicherheitslücke ausgenutzt werden kann (Exploitability, E), die Verfügbarkeit von Gegenmaßnahmen (Remediation Level , RL) und die Vertrauenswürdigkeit des Reports der Sicherheitslücke (Report Confidence, RC). Durch den Zeitbezug dieser Metriken kann die Angabe durch den Herausgeber der Bewertung der Sicherheitslücke erfolgen; es ist allerdings auch möglich, dass der Nutzer der Information diese gegebenenfalls anpasst. Eine Angabe ist hierbei nicht verpflichtend.

Die Environmental Metriken umfassen die Bewertungen des Risikos, welches von einer Sicherheitslücke ausgeht, für eine bestimmte Umgebung. Die ersten zwei Metriken dieser Gruppe umfassen das von der Sicherheitslücke ausgehende Schadenspotential (Collateral Damage Potential / CDP) und die Verbreitung der Sicherheitslücke (Target Distribution / TD). In den letzten drei Metriken werden die Anforderungen der betroffenen Systeme für Vertraulichkeit (Confidentiality Requirement, CR), Integrität (Integrity Requirement, IR) und Verfügbarkeit (Availability Requirement, AR) spezifiziert. Die Werte sind vom Benutzer anzugeben, da er das Schadenspotenzial in seiner Umgebung besser bewerten kann. Wie bei den Temporal Metriken ist die Angabe nicht verpflichtend.

Die Werte dieser Metriken werden zu einem Bewertungsvektor zusammengefasst, welcher die Eigenschaften der Sicherheitslücke beschreibt. Zusätzlich zu dem Bewertungsvektor wird eine numerische Bewertung (score) der Sicherheitslücke angegeben. Dieser Wert befindet sich auf der Skala $[0, 10]$ mit einer Schrittweite von $0, 1$. Die genaue Beschreibung der verwendeten Formeln und der verfügbaren Werte für die Metriken ist in [14] zu finden.

Einer der wichtigsten Gründe, warum CVSS beachtet werden muss, ist, dass es große Datenbanken von Sicherheitslücken gibt, die anhand dieses Schemas bewertet wurden (Beispiel NVD NIST). Die Aufteilung der Metriken in Base, Temporal und Environmental

Metriken ist vorteilhaft und führt dazu, dass der Score der Sicherheitslücken an die lokalen Gegebenheiten des IT-Systems angepasst werden kann.

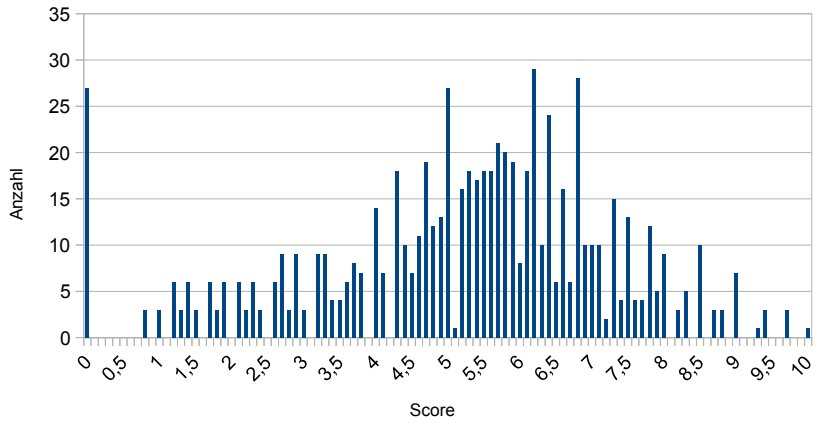


Abbildung 2.6: Verteilung von Score-Werten unter Verwendung der Base Metriken

Problematisch an CVSS ist, dass der vorhandene Wertebereich nicht optimal ausgenutzt wird. In Abbildung 2.6 wird dargestellt, wie alle möglichen Kombinationen der CVSS-Base-Vektoren auf Score-Werte abgebildet werden. Es ist deutlich zu sehen, dass nicht alle Score-Werte möglich sind, wenn nur die Base Metriken verwendet werden. Werden zusätzlich zu den Base Metriken auch Temporal und Environmental Metriken verwendet, verbessert sich die Abdeckung des Score-Wertebereiches.

Die Berechnung mit allen Metriken führt bei einer geringen Zahl von Vektoren zu einem negativen Score, wie auch in [15] festgestellt wird.

Außerdem sind die Möglichkeiten zur Anpassung der Bewertung an die lokale Umgebung über die Environmental Metriken eingeschränkt. Beispielsweise kann eine Sicherheitslücke, die in einem System die manuelle Interaktion des Nutzers benötigt, bei einer anderen Konfiguration auch ohne die Interaktion des Nutzers ausnutzbar sein.

Wie in [16] dargestellt, werden durch die eingeschränkten Werte der Metriken teilweise unterschiedlich kritische Sicherheitslücken auf den selben Score abgebildet.

Des Weiteren wird in dem offenen Brief [16] dargestellt, dass es einige Unklarheiten in den Bewertungsvorschriften gibt, welche zu unterschiedlichen Bewertungen einer Sicherheitslücke durch verschiedene Sicherheitsexperten führt.

2.5 Hidden-Markov-Modell (HMM)

Das Ziel dieses Ansatzes ist, den Sicherheitszustand, also das aktuelle Gefährdungspotential, des gesamten Netzwerkes mit Hilfe eines Hidden-Markov-Modells (HMM) zu

ermitteln.

„Ein Hidden-Markov-Modell [17] ist ein stochastisches Modell und setzt sich aus zwei separaten stochastischen Prozessen zusammen. Der versteckte (engl. hidden) Prozess ist eine Markov-Kette und besteht aus Zuständen und Übergangswahrscheinlichkeiten. Der Zustand, in dem sich dieser Prozess zum Zeitpunkt t befindet, wird mit q_t angegeben. Der äußere stochastische Prozess erzeugt zu jedem Zeitpunkt mit Wahrscheinlichkeiten belegte Ausgabesymbole. Diese Ausgabesymbole sind alles, was der Betrachter des Modells wahrnimmt.“ [18]

Um das HMM aufzustellen, wurden mehrere Annahmen getroffen. Ein Host in einem Netzwerk kann zu einem Zeitpunkt in einem der vordefinierten Sicherheitszustände sein. Die Berechnung des Host-Zustandes findet hierbei, unabhängig von eintreffenden Alarmen, in vorher definierten Zeitabständen statt. Jeder Host wird des Weiteren von mehreren Sensoren überwacht, aus deren Meldungen der Sicherheitszustand abgeleitet wird.

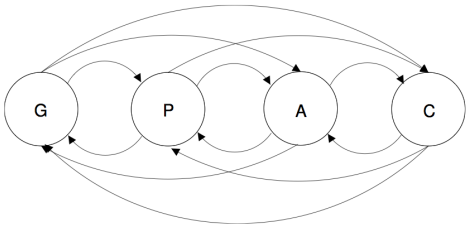
In [5] wird somit das HMM als $\lambda = (P, Q, \pi)$ definiert, mit P als „state transition probability matrix“ (Matrix der Zustandsübergangswahrscheinlichkeiten), Q als „observation probability matrix“ (Matrix der Beobachtungswahrscheinlichkeiten) und π als „initial state distribution“ (initiale Verteilung der Zustände). Die Matrix P fasst die Wahrscheinlichkeiten zusammen, dass ein Host von einem Sicherheitszustand in einen anderen wechselt. Um falsch-positive und falsch-negative Meldungen in den Eingabedaten zu erkennen, wird die „observation probability matrix“ Q aufgestellt. In welchem Zustand sich ein bestimmter Host ursprünglich befunden hatte, wird in der Verteilungsfunktion π festgehalten.

Vorgeschlagen werden z.B. folgende N Sicherheitszustände, geordnet nach Kritikalität:

- Good (G): Der Host wird nicht angegriffen.
- Probed (P): Der Host wird auf Sicherheitslücken untersucht. Dieser Zustand kann sich negativ auf die Verfügbarkeit auswirken und erhöht die Wahrscheinlichkeit eines Angriffs.
- Attacked (A): Der Host wird durch eine oder mehrere Quellen angegriffen. Das kann zu einer Senkung der Verfügbarkeit führen und die Gefahr einer Kompromittierung erhöhen.
- Compromised (C): Der Host wurde kompromittiert. Ein Verlust der Integrität, Verfügbarkeit bzw. von vertraulichen Informationen ist wahrscheinlich. [5]

In den Abbildungen 2.7(a) und 2.7(b) ist ein Beispiel für die Belegung der Matrizen P und Q unter Verwendung der vorgeschlagenen Zustände dargestellt. Die Matrizen P und Q müssen für jeden Host bzw. für Gruppen von Hosts angegeben werden.

Zunächst muss zu jedem Host auch ein Kosten-Vektor („cost vector“) C ermittelt werden,



(a) Beispiel Hidden-Markov-Modell

$$\mathbf{P} = \begin{pmatrix} p_{GG}\ p_{GP}\ p_{GA}\ p_{GC} \\ p_{PG}\ p_{PP}\ p_{PA}\ p_{PC} \\ p_{AG}\ p_{AP}\ p_{AA}\ p_{AC} \\ p_{CG}\ p_{CP}\ p_{CA}\ p_{CC} \end{pmatrix} = \begin{pmatrix} 0.95\ 0.02\ 0.02\ 0.01 \\ 0.02\ 0.95\ 0.02\ 0.01 \\ 0.02\ 0.02\ 0.94\ 0.02 \\ 0.01\ 0.01\ 0.01\ 0.97 \end{pmatrix},$$

$$\mathbf{Q} = \begin{pmatrix} q_G(1)\ q_G(2)\ q_G(3)\ q_G(4) \\ q_P(1)\ q_P(2)\ q_P(3)\ q_P(4) \\ q_A(1)\ q_A(2)\ q_A(3)\ q_A(4) \\ q_C(1)\ q_C(2)\ q_C(3)\ q_C(4) \end{pmatrix} = \begin{pmatrix} 0.7\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.7\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.7\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.7 \end{pmatrix}.$$

(b) Matrizen P und Q mit Beispielbellegung

Abbildung 2.7: Hidden-Markov-Modell [5]

um in einem so modellierten Netzwerk das Risiko für einen Host berechnen zu können. Der Kosten-Vektor gibt an, wie hoch der mögliche Schaden durch einen gegebenen Sicherheitszustand eines Hosts ist. Die Formel 2.3 beschreibt den Gesamtrisikoindex („total risk index“) $R_{h,t}$ bei N Sicherheitszuständen und der Wahrscheinlichkeit $\gamma_t(i)$, dass ein Host sich zum Zeitpunkt t in dem Sicherheitszustand i befindet.

$$R_{h,t} = \sum_{i=1}^{N} \gamma_t(i) C(i) \tag{2.3}$$

Der Risikoindex des gesamten Netzwerkes wird aus der Summe der Risikowerte der einzelnen Hosts berechnet. In einem großen Netzwerk würde ein Angriff auf wenige unbedeutende Hosts als wenig bedrohlich bewertet. Um über die Risikowerte im Allgemeinen eine Aussage treffen zu können, werden neben dem momentanen zusätzlich der durchschnittliche sowie der maximale Risikoindex berechnet.

Wird ein Alarm gemeldet, erhält er als Priorität den höchsten Risikoindex ($R_{h,t}$), der für einen der beteiligten Hosts ($h1$, $h2$) vorliegt. Dabei bezeichnen $h1$ die Quelle und $h2$ das Ziel der zu priorisierenden Meldung. Somit werden bei Hosts mit hohen Risikowerten auch die mit ihnen verbundenen Alarme hoch bewertet. Der Prioritätswert P für einen Alarm a wird mittels der Formel 2.4 berechnet.

$$P_a = \max(R_{h1,t}, R_{h2,t}) \tag{2.4}$$

Bei diesem Ansatz positiv anzumerken ist, dass beliebige Netzwerke als HMM modelliert werden können und bei Änderungen im Netzwerk die Berechnungsgrundlage nur punktuell angepasst werden muss.

Außerdem können die benötigten Matrizen und Vektoren für Gruppen von Systemen mit ähnlichen Eigenschaften definiert werden (z.B. Webserver, Laptop, Router). Deshalb ist es nicht notwendig, die Matrizen für jeden Host einzeln zu erstellen.

Durch den Vergleich des Risikoindexes, der den Prioritätswert eines Alarms angibt,

mit dem durchschnittlichen und maximalen Risikoindex, gewinnt die Aussage über die Wichtigkeit eines Alarms an Bedeutung, da sie auf quantitativen Metriken basiert und nicht z.B. auf subjektiven Meinungen von Experten. Die Verwendung von Wahrscheinlichkeiten erlaubt es, kleinere Fehler in den Messwerten der Sensoren zu tolerieren.

Es wird unter Anderem oft die positive Eigenschaft des Ansatzes betont, dass nach gewisser Zeit ohne Angriffe, der Netzwerkstatus aus dem Zustand „compromised“ ohne manuelle Anpassungen in einen weniger kritischen Zustand zurückwechseln kann. Dies wird dadurch erreicht, dass für Zeitintervalle, in denen kein Alarm vorliegt, das sogenannte „no_alert“-Ereignis, welches den minimalen Risikoindex hat, für die Berechnung verwendet wird.

Dies ist jedoch nur möglich, weil in vordefinierten und meist kleinen Zeitabständen der Risikoindex des Netzwerkes neu berechnet wird, unabhängig davon, ob ein Angriff von den Sensoren gemeldet wird. Diese Operationen sind jedoch rechenintensiv. Bei der Definition der Zeitabstände muss das Risiko, eine Situation wegen veralteter Berechnungen falsch zu bewerten, gegen die Rechenkosten und Rechenkapazität abgewogen werden.

Bei der Umsetzung sind die Autoren von [5] auf eine weitere Grenze der Rechenkapazität gestoßen. Es wurde nur ein Alarm pro Sekunde verarbeitet. Bei einem Angriff können Alarme jedoch gehäuft auftreten, was zur Folge hätte, dass viele der Alarme in eine Warteschlange eingereiht werden müssten. Um bei einem Angriff mit großen Mengen von Alarmen umgehen zu können, entschieden sich die Autoren, die Größe der Warteschlange auf 60 zu begrenzen, was jedoch zum Verlust von Ereignissen führen kann.

Ein weiterer Nachteil dieses Ansatzes ist der hohe Aufwand, der für die initiale Bestimmung der Matrizen und Kosten-Vektoren benötigt wird. Dieser Prozess kann auch nicht automatisiert werden und erfordert Expertenwissen.

Falls ein Angreifer Wissen über die Berechnungsgrundlage erlangt, ist es für ihn leicht, das System zu umgehen, indem er diejenigen Hosts angreift, die mit einem kleinen Risikoindex versehen sind. In Kombination mit dem Wissen über die begrenzte Warteschlange kann der Angreifer somit verhindern, dass sich bestimmte Alarme auf den Sicherheitszustand des Systems auswirken.

Bei der Berechnung des Risikoniveaus des Netzwerkes bemerken die Autoren des Weiteren, dass ihr Ansatz die Zusammenhänge zwischen den Hosts nicht berücksichtigt.

2.6 Attack-Graph

Nach dem Ansatz von Steven Noel und Sushil Jajodia [6] kann die Priorität eines Alarmes durch die Entfernung zwischen der angegriffenen Netzwerkkomponente und der zu

schützenden Ressource in einem Attack-Graph dargestellt werden.

Ein Attack-Graph stellt im Allgemeinen alle bekannten Pfade eines Netzwerkes dar, die ein Angreifer benutzen kann, um ein Netzwerk zu kompromittieren. Der konkrete Aufbau eines Attack-Graphen hängt stark von der Anwendung ab, in der er verwendet wird. Er lässt sich deshalb am Besten an einem Beispiel erklären. Die Abbildung 2.8(b) zeigt einen in [6] als Beispiel vorgestellten Attack-Graphen. Das zugrundeliegende Netzwerk ist in 2.8(a) abgebildet.

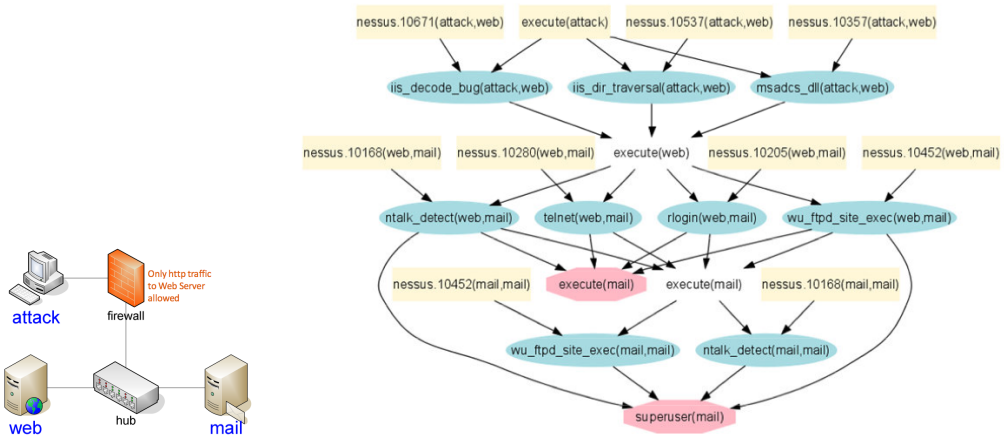
Das Beispiel-Netzwerk bestand aus einem Webserver mit einer angreifbaren Version von Microsoft Internet Information Server (IIS) und einem Mailserver, der ebenfalls eine Schwachstelle aufwies. Beide Server waren über einen Hub verbunden und durch eine Firewall geschützt, die nur Verbindungen über das HTTP-Protokoll zum Webserver zuließ.

Die im Attack-Graphen dargestellten gelben Rechtecke mit der Beschriftung *nessus.nummer (Quelle, Ziel)* stellen initiale Bedingungen dar. Genauer gesagt bedeuten sie, dass der Host *Quelle* einen Dienst mit der Sicherheitslücke *nummer* auf Host *Ziel* erreichen kann. Weiße Knoten mit der Beschriftung *execute(Host)* beschreiben die Nachbedingung, dass ein Angreifer, der diesen Knoten erreicht hat, beliebigen Code auf dem Host ausführen kann. Die blauen Ovale symbolisieren die Schritte, die ein Angreifer ausführen kann, um eine Sicherheitslücke auszunutzen. Die eingehenden Kanten verbinden die blauen Ovale mit den Vorbedingungen, die zur Durchführung eines Schrittes vorhanden sein müssen. Die ausgehende Kante zeigt auf eine Nachbedingung, die durch die erfolgreiche Ausführung des Angriffsschrittes erfüllt wird, wodurch der Angreifer weitere Möglichkeiten erhält. Die roten Achtecke schließlich stellen die kritischen Zustände dar. In dem gegebenen Beispiel war der Mailserver als kritische, zu schützende Resource markiert.

Um diesen Attack-Graphen automatisch zu generieren, wurde in [6] das TVA-Verfahren (Topological Vulnerability Analysis) eingesetzt. Dieses Verfahren untersucht das Netzwerk zuerst auf vorhandene Sicherheitslücken. In einem zweiten Schritt simuliert es mögliche Angriffe, die mit unterschiedlichen Schwachstellen starten, und zeichnet die Kombinationen der Schwachstellen auf, die zu einem erfolgreichen Angriff auf ein zu schützendes System des Netzwerkes führen können.

Um den in 2.8(b) dargestellten Attack-Graphen zu generieren, musste für das TVA-Verfahren zuerst die Möglichkeit simuliert werden, die Firewall zu durchdringen. Anschließend wurde das interne Netzwerk auf alle Möglichkeiten hin analysiert, die zur Übernahme der Kontrolle über den Mailserver führen könnten. Für diese Simulationen wird die TVA-Datenbank mit Vor- und Nachbedingungen für jede Sicherheitslücke benötigt.

Meldet ein Sensor einen Alarm, so wird dessen Priorität anhand der Entfernung auf dem kürzesten Weg zum kritischen Zustand im Attack-Graphen ermittelt. Somit würde



(a) Beispiel-Netzwerk (b) Durch TVA ermittelter Attack-Graph

Abbildung 2.8: Attack-Graph [6]

die Priorität „0“ bedeuten, dass der Angreifer nur eine Kante vom kritischen Zustand entfernt ist. Ein solcher Alarm hätte somit die höchstmögliche Priorität. Je höher der numerische Wert der Priorität ist, desto unwichtiger ist ein gegebener Alarm.

Einer der Vorteile dieses Ansatzes ist, dass bei einem mehrstufigen Angriff spätere Alarme mit einer höheren Priorität versehen werden, d.h. mit einem niedrigeren numerischen Prioritätswert. Es werden sowohl die Zusammenhänge zwischen den Alarmen als auch zwischen den Hosts berücksichtigt.

Der Attack-Graph erlaubt außerdem, eine Vorhersage über die nächsten Schritte des Angreifers zu treffen. Diese Information ist hilfreich für die Reaktion auf einen Alarm. Die Ermittlung der Prioritäten beruht auf einem einfachen Prinzip, so dass die Bedeutung der Werte leicht zu verstehen ist. Ein bedeutender Vorteil ist schließlich, dass alle Schritte dieses Ansatzes automatisiert werden können.

An dem zuvor erläuterten Beispiel lassen sich die Nachteile des Ansatzes für die Alarmpriorisierung erkennen. Größe und Komplexität des Attack-Graphen hängen direkt von der Größe des Netzwerkes ab. Um diesem Problem zu begegnen, schlagen die Autoren vor, große Netzwerke in Subnetze zu unterteilen. Dies mindert jedoch die Granularität der Prioritäten.

Zudem werden keine anderen Faktoren betrachtet, wie z. B. die Unterschiede in der Wichtigkeit der zu schützenden Objekte für die Gesamtfunktionalität des Netzwerkes, die die Granularität verbessern könnten.

Darüber hinaus ist die Erstellung des Attack-Graphen eine rechenintensive Operation. Das System kann deshalb nicht einfach auf sich kontinuierlich verändernde Netzwerke übertragen werden. Wird das Netzwerk auch nur geringfügig verändert, müssen große

Teile des Graphen angepasst werden.

Des Weiteren setzen die Autoren für die Konstruktion des Attack-Graphen eine Datenbank mit Vor- und Nachbedingungen für alle vorhandenen Sicherheitslücken voraus. Diese Datenbank, die vom TVA-Verfahren als Eingabe benötigt wird, ist jedoch nicht frei verfügbar. Der Aufwand für ihre Erstellung müsste also beim Einsatz dieses Verfahrens berücksichtigt werden.

2.7 Diskussion

Es wurden mehrere Ansätze zur Priorisierung von Alarmen vorgestellt und ihre Stärken und Schwächen aufgezeigt. In Tabelle 2.1 befindet sich eine zusammenfassende Bewertung der Ansätze anhand der folgenden Kriterien:

- Flexibilität beschreibt, wie gut ein Priorisierungssystem an lokale Gegebenheiten angepasst werden kann.
- Skalierbarkeit bezieht sich auf den Rechenaufwand, der notwendig ist, um einen einzelnen Alarm zu bewerten.
- Automatisierbarkeit bedeutet, dass ein System ohne den Eingriff von humanen Ressourcen betrieben werden kann.
- Setup bezeichnet den Aufwand, der notwendig ist, um das gegebene System in Betrieb zu nehmen. Es wird hierbei negativ bewertet, wenn Expertenwissen über das verwendete mathematische Modell benötigt wird.
- Granularität bedeutet, dass ein ausreichend großer Wertebereich für die Priorisierung verwendet und wie gut dieser ausgenutzt wird.

	RIM	FuzMet	Attack Graph	Bayes Networks	Hidden Markov	CVSS (v2)
Flexibilität	+	+	-	+	+	-
Skalierbarkeit	+	+	-	+	-	+
Automatisierbarkeit	-	-	+	o	+	-
Setup	-	-	-	-	-	o
Granularität	+	+	o	+	+	o

Tabelle 2.1: Vergleich der Ansätze

RIM ist durch die Verwendung der Indikatoren in Bezug auf die Flexibilität positiv zu bewerten. Je nach angegriffenem System werden die Indikatoren auf den Einzelfall bezogen mit Werten belegt, und durch die Verwendung der Matrizen kann die Gesamtpriorisierung an das gegebene Netzwerk angepasst werden. Da für die Berechnung der Prioritäten nur einfache Multiplikationen und Additionen verwendet werden, ist die

Skalierbarkeit positiv zu bewerten. Die Automatisierbarkeit wurde negativ bewertet, da einige der Indikatoren manuell bestimmt werden mussten. Da die Entscheidungsmatrizen von Experten erstellt und manuell an das System angepasst werden müssen, wurden die Eigenschaften des Setups negativ bewertet. Bei der Granularität erhält es eine positive Bewertung, da sehr viele Prioritätswerte möglich sind und durch die Art, wie die Entscheidungsmatrizen erstellt und validiert werden, sichergestellt ist, dass kein einzelnes Kriterium dominiert.

In FuzMet führt, wie auch bei RIM, der flexible indikatorbasierte (Metriken) Ansatz und die Anpassbarkeit durch die Fuzzy-Regeln zu einer positiven Bewertung der Flexibilität. Da die Berechnung der Priorität sehr schnell durchgeführt werden kann, ist auch die Skalierbarkeit positiv zu bewerten. Da einige der Metriken nicht automatisiert bestimmt werden können, ist die Automatisierbarkeit des Systems negativ bewertet. Das Setup ist negativ bewertet, da für den Einsatz des Systems Spezialwissen über die Fuzzy-Logic erworben werden muss und die Regeln manuell aufgestellt werden müssen. Da die von FuzMet generierte Priorität einen ausreichend großen Wertebereich aufweist, ist die Granularität positiv bewertet.

Die Priorisierung anhand des Attack-Graphen wurde in Bezug auf Flexibilität und Granularität negativ bewertet, da die Priorität an der Tiefe des Attack-Graphen gemessen wird, was entweder eine tiefe Schachtelung des Netzwerks voraussetzt oder dazu führt, dass nur wenige Prioritätswerte erreicht werden. Die Erfassung der für die Priorisierung mit Hilfe dieses Ansatzes erforderlichen Informationen ist, insbesondere bei einer steigenden Größe des Netzwerks, sehr aufwändig, und es kann keine Priorisierung erfolgen, wenn die Informationen nicht vollständig verarbeitet wurden. Ein signifikanter Vorteil dieses Ansatzes ist die Automatisierbarkeit. Da der Ansatz nach der Installation vollständig ohne menschliche Interaktion betrieben werden kann, ist er in dieser Kategorie positiv bewertet.

Weil über das im M-Correlator verwendete Bayes-Netz die Priorisierung an beliebige Netzwerke angepasst werden kann, ist es ein flexibler Ansatz. Die verwendeten Berechnungen innerhalb des Bayes-Netzwerkes können sehr schnell durchgeführt werden und führen somit zu einer positiven Bewertung der Skalierbarkeit. Da die Priorisierung an sich ohne Eingriff von Administratoren funktioniert, aber für eine Korrektur der Prioritäten ein Training des Bayes-Netzwerks und somit ein manueller Eingriff notwendig ist, wird die Automatisierbarkeit neutral bewertet. Für den Einsatz des M-Correlator-Systems und somit der Priorisierung mit Hilfe des Bayes-Netzes ist die Spezifikation der Wahrscheinlichkeitstabellen notwendig, weshalb das Setup negativ bewertet ist. Der Wertebereich zwischen 0 und 255 ist ausreichend groß gewählt und der Ansatz wird deshalb als positiv in Bezug auf die Granularität bewertet.

Da die Priorisierung anhand des CVSS-Systems nur die Bewertung von Sicherheitslücken unterstützt und auch nur eine beschränkte Anpassung des Basis-Score erlaubt,

ist die Flexibilität negativ bewertet. Die notwendigen Berechnungen für die Priorität der Alarme bei Vorhandensein einer bekannten Sicherheitslücke sind sehr einfach, weshalb die Skalierbarkeit positiv bewertet ist. Die Werte, die notwendig sind, um den Base Score über den Environmental und Temporal Score anzupassen, sind nicht automatisiert erfassbar. Das Setup für das CVSS an sich ist sehr einfach, jedoch sind die Anforderungen an das Setup der IDS-Systeme, die die Alarme liefern, sehr hoch, da sie in jedem Alarm die angegriffene Sicherheitslücke markieren müssen. Der Wertebereich, den CVSS für die Priorisierung nutzt, ist eigentlich groß genug, nur wird er nicht optimal ausgenutzt.

Die Priorisierung nach dem Hidden-Markov-Ansatz kann über die Matrizen gut an verschiedene Arten von Netzwerken angepasst werden und ist somit als flexibel zu bewerten. Negativ ist die Skalierbarkeit in Bezug auf die Anzahl der verarbeitbaren Alarme bewertet, da die Art, wie die Alarme in den Sicherheitszustand der Hosts einfließen, die Anzahl von Alarmen, die pro Minute verarbeitet werden können, beschränkt. Da das System ohne Eingriff von Menschen betrieben werden kann, ist es als automatisierbar anzusehen. Weil die Aufstellung der P- und Q-Matrizen manuell erfolgen muss und ein spezielles Expertenwissen voraussetzt, ist das Setup negativ bewertet. Dagegen kann die Granularität der Prioritäten als ausreichend angesehen werden.

Einer der größten Nachteile bei fast allen hier vorgestellten Ansätzen ist die fehlende Automatisierbarkeit. Ein Ziel dieser Arbeit ist deshalb, ein System zu gestalten, das im laufenden Betrieb ohne zusätzliches Expertenwissen auskommt.

Ein weiterer Nachteil der Verwendung von Expertenwissen ist, dass die Werte subjektiv sind und somit zu unterschiedlich guten Ergebnissen führen. Deshalb sollte bei einem für die Priorisierung von Sicherheitsalarmen eingesetzten System darauf geachtet werden, dass möglichst auf messbare Werte zugegriffen wird, die auch automatisiert erfasst werden können.

Bei fast allen Ansätzen wurde das Setup dadurch erschwert, dass bei der Konfiguration des Systems viel Expertenwissen über das eingesetzte System und die verwendeten mathematischen Methoden bzw. über das zu schützende Netzwerk notwendig ist. Deshalb ist es wichtig, die verfügbaren Informationsquellen genau zu analysieren (F.1) und diejenigen auszuwählen, die automatisiert erfasst (F.3) und kombiniert (F.2) werden können, um so möglichst ohne Expertenwissen eine ausreichende Priorisierung zu erhalten.

2.8 Anforderungen an die Priorität

In diesem Kapitel wurden mehrere Priorisierungsansätze gezeigt und deren Vor- und Nachteile vorgestellt, die zu den Bewertungen in den wichtigsten Kriterien geführt haben, wie in Tab. 2.1 abgebildet. Um die positiven Eigenschaften der Ansätze übernehmen zu können und deren Schwächen zu vermeiden, werden im Folgenden Anforderungen

an ein Priorisierungsansatz aufgestellt, die, falls erfüllt, zu positiver Bewertung in den wichtigsten Kriterien führen.

A.1 Die grundlegende Anforderung an einen Priorisierungsansatz ist, dass sie die Alarme in eine Reihenfolge bringt, die ihre tatsächliche Kritikalität für das zu schützende Netzwerk möglichst gut widerspiegelt und somit die Alarme, die den größten Schaden darstellen, am schnellsten abgearbeitet werden.

Dafür ist es wichtig, dass die so erzeugte Reihenfolge sowohl von Menschen als auch von automatisch reagierenden Systemen interpretierbar ist.

Für einen Administrator ist des Weiteren wichtig, dass die Berechnungsvorschrift, die zu dem Prioritätswert führt, transparent ist und somit eine Hilfestellung für angemessene Reaktionen bietet.

Bei der Berechnung der Priorität für Alarme muss außerdem beachtet werden, dass es eine gewisse Latenz zwischen der Erfassung von Informationen und deren Wiedergabe in der Priorität gibt. Es können beispielsweise nicht alle Flow-Informationen in Echtzeit verarbeitet werden.

A.2 Damit ein Priorisierungsansatz flexibel eingesetzt werden kann sollte er keine speziellen Anforderungen an die Netzwerkarchitektur stellen.

Mit der steigenden Verfügbarkeit von IaaS-Lösungen (Infrastructure as a Service, [19]), d. h. der Emulation einer weiteren virtuellen Netzwerkinfrastruktur-Ebene auf der tatsächlichen physischen Hardware-Ebene für Netzwerk- und Host-Systeme, gilt es, deren Unterschiede zu physikalischen Netzwerken bei der Priorisierung von Alarmen zu beachten.

A.3 Damit der Ansatz gut skaliert werden kann, ist darauf zu achten, dass der Schritt der Priorisierung selbst so wenig Zeit und Rechenleistung wie möglich benötigt. Der erste Grund dafür ist, dass eine Verzögerung in der Priorisierung auch zu einer Verzögerung der Verarbeitung des Alarms führt und somit eine notwendige Reaktion eventuell erst zu spät erfolgt. Ein Beispiel wäre das Schließen einer Firewall, das durch die Verzögerung erst nach dem Verlust der vertraulichen Informationen erfolgt. Wenn die Rechenleistung, die für die Priorisierung verwendet wird, zu hoch ist, kann dieser Umstand von einem Angreifer für einen Denial-of-Service-Angriff genutzt werden. Er könnte also mit einer hohen Anzahl wenig kritischer Alarme die rechtzeitige Verarbeitung von stark bedrohlichen Alarmen verzögern.

A.4 Die Beteiligung von Menschen an dem Prozess der Priorisierung bringt gewisse Risiken mit sich. So kann die Einschätzung einer Bedrohung von einem Experten zum anderen stark variieren, weshalb die Verwendung von subjektiven Werten minimiert werden sollte. Außerdem sollten Änderungen im Netzwerk ohne die Notwendigkeit von menschlicher Interaktion in der Priorisierung wiedergegeben

werden. Es sollte somit darauf geachtet werden, dass das System so weit wie möglich automatisierbar ist.

Wenn es jedoch nicht möglich ist, alle Eingaben für die Prioritätsberechnung automatisch und somit messbar zu erfassen, sollte darauf geachtet werden, dass die Eingabe der Informationen durch den Menschen auf einer für ihn verständlichen Ebene erfolgt und möglichst den Spielraum für Interpretationen beschränkt. Es sollte somit eine möglichst flache Lernkurve vorausgesetzt werden, um das Setup des Systems zu vereinfachen.

A.5 Bei einer Abbildung auf numerische Werte ist des Weiteren darauf zu achten, dass unterschiedliche Eingabewerte auch möglichst zu unterschiedlichen Ausgabewerten führen. Es sollte somit ein ausreichend großer Wertebereich für die numerische Priorität verwendet werden, auf den die Alarme möglichst gut verteilt werden sollten. Das Ziel ist somit eine gute Granularität der Prioritätswerte.

Eine spezielle Einschränkung der in dieser Arbeit entwickelten Priorisierung ist, dass Informationen, wie zum Beispiel die Frequenz, mit der eine Sicherheitslücke angegriffen wird, oder der Zusammenhang zwischen zwei Alarmen, nicht beachtet werden sollten, da dies von einer nachgelagerten Komponente der Software, in die dieser Priorisierungsansatz eingebaut werden soll, durchgeführt wird.

2.9 Zusammenfassung

In diesem Kapitel wurden einige Ansätze für die Priorisierung von sicherheitsrelevanten Alarmen vorgestellt und auf ihre positiven und negativen Eigenschaften hin beleuchtet. Als bedeutendste Nachteile wurden die fehlende Automatisierbarkeit, die Verwendung subjektiver Werte und die hohe Komplexität des Setups identifiziert. Diese gilt es bei der Entwicklung eines Priorisierungsprototypen zu vermeiden. Außerdem wurden Kriterien aufgestellt, die ein Priorisierungsansatz erfüllen sollte.

Kapitel 3

Analyse der Informationsquellen

Im folgenden Kapitel werden die Quellen der Informationen, die für die Priorisierung von Alarmen relevant sind, näher analysiert.

Die Informationsquellen werden hierbei in zwei Gruppen aufgeteilt. Die erste Gruppe bilden die Quellen der Strukturinformationen, die sich dadurch auszeichnen, dass sie die Struktur des zu schützenden Systems auf verschiedenen Abstraktionsebenen beschreiben. Die zweite Gruppe bilden die Quellen der Laufzeitinformationen, die einen Einblick in kurzlebige Interaktionen innerhalb des Systems geben.

Bei der Beschreibung der einzelnen Quellen wird zuerst dargestellt, welche Informationen die jeweilige Quelle zur Verfügung stellt und wie schnell sich die bereitgestellten Informationen ändern. Danach folgt eine kurze Beschreibung, wie die jeweilige Quelle zur Priorisierung der Alarme beitragen kann. Abschließend werden Möglichkeiten aufgezeigt, wie die Informationen der Quelle möglichst automatisiert erfasst werden können.

3.1 Netzwerktopologie

Als Netzwerktopologie wird im Rahmen dieser Arbeit die Informationsquelle bezeichnet, welche die Strukturinformationen über die Komponenten eines Netzwerkes und deren Kommunikationswege beinhaltet.

NT.1 Eine wichtige Information ist hierbei die Aufstellung der Netzwerke, aus denen sich das IT-System zusammensetzt. Als Netzwerke werden hierbei sowohl (ISO/OSI [20]) Layer-2-Netzwerke als auch Layer-3-Netzwerke angesehen.

NT.2 Um die Gefahr einer weiteren Verbreitung einer Infektion besser einschätzen zu können, sind die Informationen über die Verbindungen zwischen den Netzwerken wichtig.

NT.3 Da sie das primäre Ziel von Angriffen sind, sind Hosts sowie in welcher Art und Weise sie in die Netzwerke eingebunden sind, eine weitere wichtige Information.

NT.4 Eine weitere interessante Informationsquelle sind Firewalls, da sie die Kommunikationswege zwischen Netzwerken bzw. Hosts limitieren.

Die Informationen der Netzwerktopologie können bei physikalischen Netzwerken als statisch angesehen werden, da die Frequenz, mit der sie sich ändern, niedrig ist. Dies liegt daran, dass sich die Struktur der Netzwerke meist an physikalischen oder organisatorischen Strukturen, wie beispielsweise der Struktur von Gebäuden bzw. an der Erbringung von Diensten, ausrichtet. Für die Erstellung der Netzwerke sind außerdem Planungen und Vorbereitungen erforderlich, die es schwierig machen, sie dynamisch zu ändern. Bei physikalischen Netzwerken sind diese Vorbereitungen noch höher, da eine Änderung meist einen Eingriff in die physikalischen Strukturen, also Verkabelungen und Standorte von Systemen notwendig machen. Des Weiteren muss bei jeder Änderung eines Netzwerkes, auch wenn es nur die Anpassung der Verbindung zweier Netzwerke ist, die Gefahr eines Ausfalls beachtet werden, welcher hohe Kosten durch verlorene Arbeitszeiten erzeugen kann. Obwohl virtuelle Netzwerke sehr schnell aufgebaut und verändert werden können, werden sie meist für die Erbringung eines Dienstes erstellt, dessen Lebensdauer von der Umsetzung bis hin zur Stilllegung Wochen bis Monate in Anspruch nehmen kann.

Die Netzwerktopologie stellt eine wichtige Informationsquelle für die korrekte Priorisierung von Alarmen dar. Mit Hilfe der aus ihr erlangten Informationen können beispielsweise Aussagen darüber getroffen werden, auf welchen Wegen sich ein Angreifer durch das Netzwerk bewegen kann und ob wichtige zu schützende Ziele in Reichweite des angegriffenen Hosts sind.

Somit ist die Priorität eines Alarms, die einen Host betrifft, der eine zentrale Stellung im Netzwerk bzw. viele andere Hosts in seiner Reichweite hat, höher zu bewerten als der gleiche Angriff auf ein gut isoliertes System. Wie gut ein System von anderen isoliert ist, kann mittels der Informationen über NT.1 und NT.3 festgestellt werden.

Hierbei spielt die Art des Netzwerkes in Bezug auf die Schichten des ISO-Modells eine wichtige Rolle. Die Isolation mehrerer Hosts, die über ein Netzwerk auf Layer-2 verbunden sind, ist meist geringer, als wenn die selben Hosts über ein oder mehrere Layer-3-Netzwerke miteinander verbunden sind. Sind mehrere Hosts mit einem Layer-2-Netzwerk verbunden, so ist ein Alarm, der einen dieser Hosts betrifft, höher zu bewerten als der gleiche Alarm, wenn die Hosts mittels eines Layer-3-Netzwerkes verbunden sind. (NT.1, NT.2 und NT.3)

Eine weitere wichtige Information, die aus der Netzwerktopologie gewonnen werden kann, ist die Position des Angreifers relativ zum zu schützenden Netzwerk. Einem externen Angreifer stehen meist mehrere Sicherheitsvorkehrungen im Weg, die er erst überwinden muss, um nachhaltigen Schaden im Zielsystem anzurichten. Befindet sich der

Angreifer jedoch bereits im zu schützenden Netzwerk, sind die ihm im Weg stehenden Schutzmaßnahmen und die Entfernung zu wertvollen Ressourcen des Unternehmens meist geringer. Somit wäre ein Angriff, der durch einen internen Angreifer ausgeführt wird, höher zu bewerten als der gleiche Angriff von einer externen Quelle aus. (NT.2)

Es gibt verschiedene Möglichkeiten, wie Topologie-Informationen gewonnen werden können. In gut organisierten Netzwerken werden Tools eingesetzt, um Netzwerk-Pläne der Netzwerktopologie vor der Installation zu erstellen. Da diese Pläne jedoch mit der Zeit veralten können und somit nicht mehr der realen Netzwerktopologie entsprechen, ist es besser, Tools einzusetzen, die die Struktur des Netzwerkes automatisiert erfassen.

Ein Tool für die automatische Erfassung der Netzwerktopologie ist der Security Scanner *NMAP* (Network Mapper) [21]. Dieser erfasst durch verschiedene Techniken die Netzwerktopologie automatisiert und kann sie in einem XML-Format exportieren, das dann für die Rekonstruktion der Netzwerktopologie verwendet werden kann. Da es *NMAP* teilweise nicht möglich ist, verschiedene Netzwerk-Interfaces einem System zuzuordnen, ist es sinnvoll die von ihm generierten Informationen mit Informationen der Hosts anzureichern, um so die Zusammenhänge zwischen Hosts und Netzwerk-Interfaces herzustellen. Außerdem kann es in sicheren Netzwerken notwendig sein, dass der Scan von mehreren Hosts aus erfolgen muss, da durch Sicherheitseinrichtungen wie Firewalls oder durch den Einsatz von Network Address Translation (NAT) die Sichtweite von *NMAP* eingeschränkt sein kann.

Über *NMAP* kann somit das Vorhandensein von Hosts innerhalb des Netzwerkes festgestellt werden und wie diese in die Netzwerke eingebunden sind (NT.3). Ein *NMAP*-Scan kann außerdem über Layer-3-Netzwerkgrenzen hinweg Hosts erkennen. Wird ein Scan von mehreren Quellen aus ausgeführt, ist es somit möglich die Informationen über das Routing des Netzwerkes zu erhalten (NT.2). Aus den unterschiedlichen Scan-Ergebnissen des selben Ziels von mehreren Quellen aus lassen sich auch Effekte von Firewalls nachweisen (NT.4). Werden die so gewonnenen Informationen über die Hosts zusammengeführt, ist es möglich die einzelnen Layer-3-Netzwerke zu erkennen. Um Informationen über die Layer-2-Netzwerke zu erhalten, müssen die Ergebnisse des *NMAP*-Scans mit den Host-Informationen zusammengeführt werden (NT.1).

Ein weiteres Tool für die automatisierte Erkennung der Netzwerktopologie ist der von OpenNMS [22] verwendete Linkd [23]. Dieser sammelt die Informationen über die Netzwerktopologie über verschiedene Protokolle direkt von den Netzwerkkomponenten.

Wird eine IaaS-Lösung eingesetzt, ist es sinnvoll, die physikalische Hardware mit den oben beschriebenen Tools zu erfassen, wohingegen die Informationen über die virtualisierte Hardware besser direkt aus der verwendeten Infrastruktur-Lösung (wie z.B. OpenStack [24]) bezogen werden sollten, da so die für die Erfassung der Topologie verwendeten Ressourcen reduziert werden können. Auf diesem Weg können die Informationen über NT.1, NT.2 und NT.3 bestimmt werden.

3.2 Dienststruktur

Eine weitere für die Priorisierung relevante Informationsquelle ist die Dienststruktur, die die Strukturinformationen der im Netzwerk vorhandenen Dienste und deren Abhängigkeiten untereinander umfasst. Unter einem Dienst wird hierbei eine Einheit verstanden, die eine bestimmte Funktionalität anbietet und gegebenenfalls andere Dienste für die Erbringung ihrer Funktionalität benötigt.

Analog zur Netzwerktopologie kann die Dienststruktur als eine statische Information angesehen werden, da ein Wegfallen und Hinzukommen von Diensten Vorbereitungen und Aufwände innerhalb eines Unternehmen, wie Einarbeitung der zuständigen Administratoren, die Planung und Bereitstellung bzw. Installation der Hard- und Software für die Erbringung des Dienstes, notwendig machen.

DS.1 Über die Verbindungen der Dienste untereinander können weitere Informationen abgeleitet werden, wie zum Beispiel eine Vorhersage über die Gefährdung der Verfügbarkeit eines Dienstes im Falle des Ausfalles eines anderen Dienstes.

DS.2 Eine weitere wichtige Information ist, wie sich der Wert eines Dienstes in Bezug Vertraulichkeit, Integrität und Verfügbarkeit verhält.

DS.3 Zu jedem Dienst können die Netzwerkexperten Informationen hinterlegen, die sich auf die Wichtigkeit des Dienstes beziehen.

Meist kann ein Angriff auf eine Komponente des Netzwerkes auf einen Dienst abgebildet werden. Dadurch können die Informationen, die aus der Dienststruktur gewonnen werden, in die Priorität des Alarms einfließen. Ähnlich wie bei der Netzwerktopologie spielt bei der Priorität des Alarms die Zentralität eines Dienstes, also wie viele andere Dienste von ihm abhängen, eine wichtige Rolle. Der Ausfall eines zentralen Dienstes kann viele andere Dienste in Mitleidenschaft ziehen. Je nach Verwendung kann beispielsweise ein DNS-Dienst zu einem zentralen Dienst werden.

Eine Möglichkeit, die Informationen der Dienststruktur in Bezug auf DS.1 für ein bestehenden Netzwerk zu generieren, ist das Einlesen der Konfigurationsdateien aller installierten Anwendungen. Da jedoch die Struktur der Konfigurationsdateien und die Art, wie beispielsweise Applikationsserver mit Datenbanken usw. verknüpft werden, von Anwendung zu Anwendung variieren, ist es einfacher diese Informationen durch die manuelle Erstellung des Dienstgraphen zu erfassen. Hierbei können Informationen, die durch die Analyse der Netzwerktopologie erfasst wurden, helfen, die Korrektheit der Dienststruktur zu validieren. Die erfasste Dienststruktur kann außerdem für ein Monitoring des Netzwerkes genutzt werden. Um das Einlesen der Dienststruktur über Servergrenzen hinweg zu vereinfachen, ist es sinnvoll, ein Tool für das zentrale Management der Konfigurationen von Servern zu verwenden, z.B. Chef [25], Puppet [26].

Die Bewertung des Dienstes in Bezug auf Vertraulichkeit, Integrität, Verfügbarkeit

und somit DS.2 kann auf unterschiedliche Arten bestimmt werden. Wird ein System eingesetzt, das die Klassifikation der Vertraulichkeit der von einem Dienst verwendeten Daten erlaubt, kann diese Information auch dazu verwendet werden, den Dienst in Bezug auf Vertraulichkeit zu klassifizieren. Für die Bestimmung des Wertes der Integrität könnten beispielsweise Informationen darüber verwendet werden, wie lange es dauert, eine Zerstörung der Daten zu erkennen und zu beheben bzw. welche Kosten damit verbunden sind. Die Informationen aus Messungen, wie häufig ein Dienst genutzt wird, können dazu verwendet werden, die Wichtigkeit eines Dienstes in Bezug auf seine Verfügbarkeit zu bestimmen.

Die Informationen für DS.3 könnten beispielsweise darüber ermittelt werden, wie teuer ein Ausfall des gegebenen Dienstes ist. Es ist hierbei jedoch nicht sinnvoll, diesen Wert für jeden elementaren Dienst zu spezifizieren, sondern auf einer höheren Ebene. Beispielsweise wäre es nicht zielführend, den Wert eines HTTP-Reverse-Proxy zu spezifizieren, sondern die Dienste, die über ihn erreichbar sind und ihn somit voraussetzen.

3.3 Host

Dienste bieten eine höhere Abstraktionsebene für die Definition, welche Funktionalitäten im Netzwerk benötigt werden. Sie erbringen die Funktionalität aber nicht selbst. Deshalb müssen sie auf Hosts abgebildet werden, auf denen dann die Software läuft, um den Dienst zu erbringen. Da die Hosts das Hauptziel der Angriffe darstellen, sind sie eine wichtige Informationsquelle für die Priorisierung der Alarme. Klassifizieren lassen sich die über den Host gewonnenen Informationen als Strukturinformationen auf einer niedrigeren Abstraktionsebene als die Dienststruktur und die Netzwerktopologie.

Die Informationen der in Bezug auf einen Host relevanten Informationen sind kurzlebiger als die der Dienste und Netzwerktopologie. Abhängig von der Art des Systems (Server, Laptop) ändert sich beispielsweise die Art der Einbindung in Netzwerke entweder so gut wie gar nicht bzw. im Rahmen von einigen Stunden. Informationen über die auf den Systemen verwendete Software ändert sich meist innerhalb von Tagen bzw. Wochen (z.B. Microsoft Update [27]).

H.1 Um eine Verbindung zwischen der Netzwerktopologie und den Informationen des Hosts herstellen zu können, ist es wichtig, zu erkennen, wie genau ein Host in ein Netzwerk eingebunden ist. Dies betrifft sowohl die verfügbaren Layer-2- als auch Layer-3-Netzwerke.

H.2 Die Anzahl der von einem Host bereitgestellten Dienste ist eine wichtige Information, um die Priorität des Hosts innerhalb des Netzwerkes einschätzen zu können.

H.3 Eine wichtige Information sind die vom Host bereitgestellten Ports. Über die Ports

kann im Falle eines eintreffenden Alarms festgestellt werden, ob die angegriffene Software bzw. der Port des Hosts im Augenblick des Angriffs aktiv war. Dies kann eine wichtige Information für die Priorisierung von Alarmen sein, da so die Erfolgschancen eines Angriffs besser eingeschätzt werden können.

H.4 Für die Erbringung eines Dienstes ist immer auch die dazugehörige Software notwendig. Die Information über die Software kann dazu verwendet werden, um Sicherheitslücken in Diensten zu erkennen und eine bessere Einschätzung über die Erfolgschancen eines Angriffs auf eine bestimmte Sicherheitslücke geben zu können.

H.5 Der Erfolg eines Angriffs hängt teilweise von speziellen Anforderungen bzw. Konfigurationen einer Software auf einem System ab. Es ist deshalb wichtig auf dem Host zu erkennen, ob die Voraussetzungen für einen erfolgreichen Angriff gegeben sind oder nicht.

H.6 Für manche Angriffe, die auf das Überlasten eines Systems ausgelegt sind, spielen die auf den Dienst bereitstellenden Hosts verfügbaren Ressourcen und deren Auslastung eine wichtige Rolle.

H.7 Auf den Hosts können außerdem Informationen darüber erfasst werden, welche Benutzer aktuell aktiv mit ihnen verbunden sind. Dadurch ist es möglich, die Priorität eines Hosts innerhalb eines Netzwerkes besser einschätzen zu können, da ein aktiv genutztes System wichtiger ist, als ein zur Zeit nicht verwendetes.

Die Informationen über die von einem Host aus erreichbaren Netzwerke lassen sich beispielsweise auf Linux-Systemen mit den Tools *ip* bzw. *ifconfig* identifizieren. Auf diese Weise lassen sich Informationen über H.1 automatisiert erfassen.

Ist die Position eines Hosts im Netzwerk gefunden, so ist es interessant, die von ihm bereitgestellten Netzwerkdienste zu analysieren. Auf Linux-Systemen kann hierfür das *netstat* eingesetzt werden. Mittels dieses Tools ist es möglich, eine Zuordnung zwischen IP-Adressen, geöffneten Ports und den dahinter liegenden Prozessen herzustellen. Es kann somit die Brücke zwischen H.3, H.1, H.2 und H.4 geschlagen werden.

Die über *netstat* erhaltene Information über die Prozesse, kann mit Hilfe des „/proc“ Dateisystems unter Linux zur Analyse der Interna des Prozesses und insbesondere der dahinter liegenden Software verwendet werden.

Um die Abhängigkeiten zwischen den verwendeten Bibliotheken zu analysieren, ist *ldd* nützlich, da es dazu verwendet werden kann, die Struktur der ausführbaren Dateien bzw. Bibliotheken zu analysieren. Bei der Analyse der Anwendungen muss natürlich auch darauf geachtet werden, dass es je nachdem, welche Frameworks bzw. Programmiersprachen verwendet werden, auch weitere Ebenen geben kann, die es zu analysieren gilt. Ein Beispiel für solch eine weitere Ebene existiert bei *Java*-Anwendungen. Auf der Ebene der nativen Bibliotheken sehen *Java*-Anwendungen meist sehr ähnlich aus, da

sie auf der selben *Java Virtual Machine* aufbauen. Betrachtet man aber die eingesetzten Java-Bibliotheken, können Sicherheitslücken, die auf der nativen Ebene nicht sichtbar sind, erkennbar werden. Für Java müsste deshalb *jar* zur Analyse der Bibliotheken eingesetzt werden bzw. die Debugging- und Management-Schnittstellen der *JVM* (z.B. *JMX*). Für andere Frameworks existieren ähnliche Tools zur Analyse der Anwendungen.

Die Information, welche Nutzer gerade aktiv ein System verwenden H.7, müsste von den Diensten selbst bereit gestellt werden, was eine solche Erfassung schwierig in der Umsetzung macht, aber durchaus denkbar ist. Ein Beispiel ist das *Linux*-Kommando *who*, welches die aktuell in einem *Linux*-System eingeloggten Nutzer anzeigt. Ein weiteres nützliches Beispiel ist das von dem *IMAP*-Server *dovecot* bereitgestellte *doveadm who*, welches alle im Augenblick mit dem Server verbundenen Nutzer auflistet.

Die für die Sammlung der Informationen notwendigen Tools können sich zwar je nach Betriebssystem stark unterscheiden, jedoch können die Informationen immer automatisiert erhoben werden.

Um die Informationen für H.5 zu erfassen, wäre es möglich, Konfigurationsdateien zu analysieren, die für die Konfiguration der Dienste verwendet wurden. Bei *Linux*-Systemen liegen diese meist zentral innerhalb des ‘/etc“ Verzeichnisses. Da sich die Struktur und der Inhalt der Konfigurationsdateien oft stark unterscheiden, ist es nur mit hohem Aufwand möglich, die Informationen über die bereitgestellten Dienste auf diesem Weg zu erhalten. Die Automatisierung dieses Prozesses ist jedoch durchaus denkbar.

Des Weiteren kann die Erkennung der Sicherheitslücken H.4, der bereitgestellten Dienste H.2 und die Einbindung in das Netzwerk H.1 mittels spezieller Software, den sogenannten Netzwerkscannern, wie beispielsweise dem bereits in 3.1 vorgestellten *NMAP*, *OpenVAS* (open source [28]) oder Nessus (kommerziell [29]) durchgeführt werden. *OpenVAS* bietet eine große Auswahl an Testangriffen und Prüfungen, sogenannter Network Vulnerability Tests (NVT), die einzeln oder in größeren, thematisch gegliederten Gruppen in dem zu prüfenden Netzwerk durchgeführt werden. Welche Tests genau gegen welche Hosts oder in welchem Modus (sicher - ohne den Zielhost zu beeinflussen, aggressiv - mit allen Mitteln) von *OpenVAS* ausgeführt werden, wird von einem Experten mit Hilfe eines *OpenVAS*-Client in einem Scanauftrag spezifiziert. Sind die Rahmenbedingungen des Scanauftrags einmal festgelegt kann dieser automatisiert wiederholt werden.

Die Ergebnisse eines Scanauftrags werden von *OpenVAS* in einem Bericht festgehalten und in einer internen Datenbank abgelegt. Dabei werden die gefundenen Sicherheitslücken und Schwachstellen aufgelistet, sowie eine Abschätzung der Gefahr, die diese für das Netzwerk darstellen.

Werden die Scans mehr als einmal durchgeführt, so enthalten die Berichte die Angaben zu dem sogenannten Trend, ob sich die Sicherheit innerhalb des Netzwerkes verbessert

bzw. verschlechtert hat [30]. Es kann auch spezifiziert werden, in welchem Format die Berichte gespeichert werden, wie z. B. in .csv-, .txt- oder .pdf-Dateien. Für eine automatisierte bzw. maschinelle Verarbeitung eignet sich jedoch die Repräsentation der Daten im XML-Format am Besten.

Auf den Hosts lassen sich im Prinzip die Auslastung und die zur Verfügung stehenden Ressourcen des Systems bestimmen (H.6). Diese Information kann dafür verwendet werden, um Angriffe besser priorisieren zu können, deren Ziel es ist, die Zielsysteme zu überlasten (Denial of Service). Der Gedanke hierbei ist, dass ein System, das noch weit von seiner maximalen Belastbarkeit entfernt ist, einen solchen Angriff ohne Beeinträchtigung für die normalen Nutzer überstehen kann. Befindet sich das System jedoch nahe der maximalen Auslastung, ist es sehr wahrscheinlich, dass der Angriff erfolgreich verläuft. Deshalb sollte in diesem Fall ein Alarm höher priorisiert werden. Die Erfassung von Echtzeitinformationen über die aktuelle Auslastung ist schwierig umzusetzen und mit hohen Kosten verbunden. Auf *Linux*-Systemen könnte beispielsweise die Ausgabe des Befehls *top* dazu verwendet werden. Außerdem werden diese Informationen von IDS-Systemen dazu verwendet, um Sicherheitsalarme zu generieren. Sollte die Auslastung des Hosts einer gewissen Regelmäßigkeit unterworfen sein, wäre es jedoch möglich, von der Verwendung der Echtzeitinformationen abzusehen und eine angenäherte Funktion für die Bestimmung der „normalen“ Auslastung zu einem gegebenen Zeitpunkt zu verwenden.

Für die Sammlung der meisten Informationen muss also eine spezielle Software auf dem Host installiert sein, die es ermöglicht, ihn von innen heraus zu untersuchen, wobei einige der Informationen auch von außen aquirierbar sind.

3.4 Flow-Daten

Hosts bieten eine gute Strukturinformation über die Interna der im Netzwerk vorhandenen Systeme. Wie und wie oft die Systeme miteinander kommunizieren, ist für die Priorisierung von Alarmen ebenfalls von Bedeutung. Diese Laufzeitinformationen können in Form von Flow-Daten erhoben werden. Als Flow wird dabei ein Datenstrom bezeichnet, bei dem die Datenpakete ähnliche Eigenschaften haben, z. B. gleiche Ziel- oder Quelladresse [31]. Flow-Daten enthalten hierbei typischerweise:

FD.1 die Quelle des Datenflusses

FD.2 das Ziel des Informationsflusses

FD.3 die Zeitpunkte des Beginns und Endes des Informationsflusses

FD.4 die Menge der Daten, die zwischen den Systemen ausgetauscht wurden

FD.5 die Art der Kommunikation, die stattgefunden hat, bzw. das verwendete Protokoll

Die rohen Flow-Daten zwischen Systemen sind zwar eine sehr kurzlebige Informationsquelle, da Verbindungen mit einer sehr hohen Frequenz aufgebaut bzw. geschlossen werden. Jedoch weisen die für die Priorisierung relevanten Informationen, also welche Kommunikationswege existieren bzw. verwendet wurden, eine viel geringere Änderungsrate auf.

Die Flow-Daten können zur richtigen Priorisierung von Sicherheitsalarmen beitragen und helfen, Aussagen über die Wichtigkeit der beteiligten Hosts zu treffen.

Tritt ein Host oft als Ziel (FD.2) von Verbindungen auf, so sind Angriffe auf ihn mit höherer Priorität zu bewerten, da er von vielen anderen Systemen benötigt wird.

Je mehr Verbindungen ein Host zu anderen Hosts aufgebaut hat, desto wichtiger ist ein Angriff auf ihn, da ggf. mehr Informationen durch Folgeangriffe gefährdet sind (FD.1). Der Host hat somit eine höhere Reichweite innerhalb des Netzwerkes und die Informationen, auf die zugegriffen wurde, bzw. Zugangsdaten zu ihnen, sind wahrscheinlich über den Host verfügbar.

Außerdem kann über die Flow-Daten FD.1 - FD.5 festgestellt werden, ob ein Host nach Eintreffen eines Sicherheitsalarms verdächtige Kommunikationsvorgänge gestartet hat. Diese Information würde zu einer höheren Priorisierung des Alarms führen, da sie als Indiz für eine Kompromittierung des Systems gewertet werden kann. Außerdem können wichtige Kommunikationswege identifiziert werden, die durch einen Angriff gefährdet werden könnten. Ein Angriff auf zentrale Komponenten dieser Kommunikationswege kann dadurch höher bewertet werden.

Die automatische Erfassung der Informationen FD.1 - FD.5 kann verteilt im Netzwerk auf den einzelnen Hosts bzw. auf den zentralen Knoten, die zwischen den einzelnen Netzwerken vermitteln, erfolgen. Hierbei muss jedoch die Sichtbarkeit des Datenverkehrs beachtet werden. Ein Beispiel einer Software, die diese Erfassung der Daten durchführen kann, ist *Vermont* (Versatile Monitoring Toolkit). Entwickelt als eine „Referenzimplementierung der *IETF* [Internet Engineering Task Force] für die Protokolle *IPFIX* und *PSAMP* im Rahmen des Projektes *HISTORY* (High Speed Network Monitoring and Analysis) ermöglicht *Vermont* „die Aufzeichnung und Aggregation von Paketen im Gigabit-Bereich mit Standard-PCs“ [31]. Die Protokolle *IPFIX* (Internet Protocol Flow Information Export) und *PSAMP* (Packet Sampling) spezifizieren, in welcher Form die Internet-Flow- bzw. Packet-Informationen bei einer Abtastung eines Datentransfers dargestellt sein müssen, und sie stellen Standards zum Austausch solcher Informationen dar ([32], [33]). *Vermont* kann in den Modus „IPFIX Probe“ versetzt werden, in dem es die Laufzeitinformationen beinhaltet und diese in *IPFIX*-konformer Form an das System schickt, das die Daten benötigt.

3.5 IDS-Alarme

Die wichtigste Informationsquelle für die Priorisierung ist natürlich der Alarm selbst. Er trägt die Laufzeitinformationen über einen festgestellten Angriff auf ein zu schützendes System. Diese sind unter anderem:

AL.1 Der Zeitpunkt, an dem der vermeintliche Angriff stattgefunden hat.

AL.2 Die Quellen und das Ziel des Angriffes in unterschiedlichen Granularitätstufen, wie beispielsweise IP-Adresse, Host, Port oder/und der angegriffene Prozess.

AL.3 Welche Sicherheitslücke im Angriff verwendet wurde.

AL.4 Welcher Sensor den Angriff erkannt hat und wie dieser die Gefahr initial bewertet.

IDS-Alarme sind bezogen auf die Priorisierung als eine statische Informationsquelle anzusehen. Alarme können zwar mit einer sehr hohen Rate erzeugt auftreten, jedoch verändern sich die für die Priorisierung relevanten Informationen nach der Erzeugung des Alarms nicht mehr.

Ausgehend von diesen Basisinformationen können weitere Informationen abgeleitet werden, zum Beispiel der Sicherheitstatus der einzelnen Netzwerkkomponenten und der Zustand des Netzwerkes im Allgemeinen (wie beispielsweise in [5]). Außerdem werden die Attribute des Alarms dazu verwendet, für die Priorisierung relevante Informationen der anderen Informationsquellen abzurufen.

Die Informationen über Alarme werden durch *Intrusion-Detection-Systeme* (IDS) erfasst und üblicherweise im *Intrusion-Detection-Message-Exchange-Format* (IDMEF) an das verarbeitende System weitergeleitet. Dabei werden die Attribute des Alarmes im Format XML (Extensible Markup Language) mit Hilfe bestimmter Tags in einem sowohl maschinen- als auch menschenlesbaren Dokument zusammengefasst. Wie in [7] beschrieben besitzt IDMEF ein objektorientiertes Modell der Daten, die für einen Alarm gesammelt werden können, wodurch es erweiterbar und für ein IDS anpassbar ist. An der Spitze des Datenmodells steht die Oberklasse **IDMEF-Message**, von der zwei Klassen abgeleitet sind: **Alert** und **Heartbeat**. Es soll die Klasse **Alert** betrachtet werden, da sie die Repräsentation eines oder mehrerer Ereignisse ist, die von einem IDS registriert und als bedrohlich eingestuft wurden. Die **Alert** Klasse besitzt einige Attribute, die eine Struktur zum Speichern der gesammelten Daten vorgeben. Die Attribute sind z.B.: die Zeit, wann ein Ereignis registriert und der Alarm dazu erzeugt wurde, von welchem Sensor bzw. Analyzer des IDS ein Alarm gemeldet wurde, die Daten über den Quell- und Ziel-Host, die Abschätzung der Bedrohung und somit der Priorität des Alarms, zusätzliche Daten und die Angabe der in der Liste der Common Vulnerabilities and Exposures (CVE) dafür vergebenen Kennung. Die *CVE-ID* ist eine standardisierte, eindeutige Bezeichnung einer Sicherheitslücke, mit der detaillierte Informationen über diese Sicherheitslücke aus einer der Schwachstellendatenbanken abgefragt werden

können. [34]

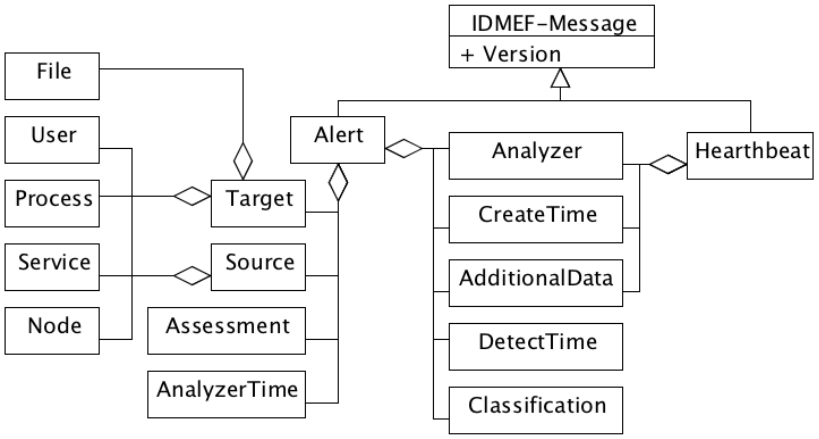


Abbildung 3.1: IDMEF-Modell nach [7]

Die Abbildung 3.1 zeigt das IDMEF-Datenmodell.

Somit steht die Information über AL.1 in dem Feld für **DetectTime**. Die Informationen für AL.2 sind in den Attributen des Alarms, die die Klassen **Source** und **Target** erfassen, meist in der Form einer **Node**, das bedeutet Netzwerkkomponente, enthalten. Die angegriffene Sicherheitslücke AL.3 wird in IDMEF über die Klasse **Classification** abgebildet. Der Sensor, der die Anomalie erkannt hat, kann durch das Attribut **analyzerid** der Klasse **Analyzer** identifiziert werden.

3.6 Schwachstellendatenbanken

Da die reine Benennung der angegriffenen Sicherheitslücke durch das IDS noch keine Auskunft über das tatsächliche Ausmaß der Sicherheitslücke und der von ihr ausgehenden Gefahren für das Netzwerk gibt, ist es notwendig, auf Schwachstellendatenbanken zuzugreifen. Informationen, die durch die Schwachstellendatenkbanken bereitgestellt werden können, sind:

- Ein übergreifendes Kennzeichen, über das Informationen aus verschiedenen Schwachstellendatenbanken verbunden werden können.
- Eine für Administratoren gedachte textuelle Beschreibung der Sicherheitslücke.
- Die Gefährdung der Vertraulichkeit, Integrität und Verfügbarkeit des Systems, die durch die Ausnutzung der Schwachstelle entstehen kann.
- Eine Liste der Anwendungen bzw. Bibliotheken, die von der Schwachstelle betroffen sind.

- Die Angabe der Bedingungen, unter welchen es möglich ist, die Schwachstelle auszunutzen.
- Eine Schätzung, wie wahrscheinlich eine erfolgreiche Ausnutzung der Schwachstelle ist.
- Möglichkeiten, wie die Schwachstellen miteinander kombiniert werden können, um einen erfolgreichen Angriff durchzuführen.
- Die Wahrscheinlichkeit der Übernahme der Kontrolle über den Zielhost durch die gegebene Sicherheitslücke.

Betrachtet man die Änderungsrate einer Schwachstellendatenbank bezogen auf jede existierende Software, so ist die Änderungsrate als sehr hoch anzusehen. Da für ein Unternehmen jedoch nur die Schwachstellen relevant sind, die die tatsächlich eingesetzte Software betreffen, ist die Änderungsrate der relevanten Information nicht so hoch einzustufen.

Ist es dem System möglich, einem Alarm eine ausgenutzte Schwachstelle zuzuordnen, können die Informationen der Sicherheitslücke direkt in die Priorisierung des Ereignisses einfließen. Beispielsweise ist ein Alarm, der eine Sicherheitslücke betrifft, die einen starken negativen Einfluss auf Vertraulichkeit, Integrität und Verfügbarkeit hat, wichtiger, als ein Alarm der sich auf eine Schwachstelle mit nur begrenzter negativer Auswirkung bezieht. Das gilt natürlich auch für die Erfolgswahrscheinlichkeit des Angriffs, die aus den Schwachstelleninformationen hervorgeht. Ist die Feststellung der Sicherheitslücke nicht möglich, können die Informationen allerdings auch indirekt in die Priorität einfließen, indem sie mit anderen Informationsquellen kombiniert werden. Möglichkeiten, wie diese Kombination der Informationsquellen aussehen könnte, werden im nächsten Abschnitt genauer betrachtet.

Es gibt mehrere Datenbanken, die von Behörden bzw. Unternehmen gepflegt und veröffentlicht werden. Ein Beispiel ist die National Vulnerability Database (NVD), die vom National Institute of Standards and Technology, welches wiederum ein Teil des U.S. Department of Commerce ist, gepflegt wird. Die Daten über die Sicherheitslücken werden von der NVD in Form einer XML-Datei zum initialen Herunterladen angeboten. Damit die Informationen aktuell bleiben, gibt es zusätzlich die Möglichkeit, die Veränderungen mit Hilfe von *RSS-Feeds* automatisiert zu erkennen.

Zur Strukturierung der Informationen über eine Sicherheitslücke wurde von der *NVD* ein XML-Schema festgelegt. Die Version 2.0 kann unter der *URL* [35] abgerufen werden. Dabei stellt das Element „nvd:entry“ einen Auszug der Datenbank dar, der eine Liste mit den Sicherheitslücken beinhaltet. Zu jeder von ihnen werden eine Reihe von Informationen gespeichert, wie z. B. die *CVE-ID*, wann die Sicherheitslücke entdeckt oder veröffentlicht wurde, eine Liste der betroffenen Softwareprodukte, eine textuelle Beschreibung und die CVSS-Einstufung der Sicherheitslücke, die bereits in 2.4 beschrieben

wurde.

3.7 Kombination der Informationsquellen

Jede der Informationsquellen alleine liefert bereits wertvolle Informationen für die Priorisierung von Alarmen. Durch die Kombination der Informationsquellen ist es jedoch möglich, die Priorisierung weiter zu verbessern. Deshalb werden nun einige Kombinationen aus den oben vorgestellten Informationsquellen aufgelistet und deren Verwendung bei der Ermittlung der Alarmpriorität erläutert.

Jeder Dienst wird von einem Netzwerkexperten klassifiziert. Da jeder Dienst von Hosts erbracht wird, ist es möglich, diese Dienstklassifizierung auf die Hosts zu übertragen und so die Wichtigkeit der Hosts untereinander besser einschätzen zu können. Diese abgeleitete Klassifizierung kann dann als Teil der Prioritätsberechnung verwendet werden. Wenn die Vertraulichkeit, Integrität und Verfügbarkeit der Dienste, die von einem Host erbracht werden, hoch eingestuft wurden, ist ein Alarm, der ihn als Quelle oder Ziel referenziert, höher einzustufen als einer, der sich nur auf niedrig eingestufte Hosts bezieht.

Die auf den Hosts erfassten Informationen darüber, welche Benutzer aktuell aktiv mit ihnen verbunden sind, kann mittels der Verbindung von Hosts zu bereitgestellten Diensten auf die Dienststruktur abgebildet werden. Je nach Ausrichtung des Unternehmens ist es sehr wahrscheinlich, dass der Schaden, der durch den Ausfall eines Dienstes entsteht, direkt proportional zu der Art und Anzahl der Nutzer ist, die den Dienst aktuell verwenden. Somit ist ein Dienst, der von vielen wichtigen Personen im Unternehmen verwendet wird, als wichtiger anzusehen als einer, der nur selten bzw. von weniger wichtigen Personen verwendet wird.

Wird die so generierte Einstufung der Wichtigkeit des Hosts mit der Netzwerktopologie kombiniert, ist es auf diesem Weg möglich, ein Ranking der Netzwerke zu erstellen, welches wiederum in der Priorisierung verwendet werden kann. Das bedeutet, ein Host, der an sich keine Dienste bereitstellt, kann so ein höheres Ranking erhalten, wenn er Teil eines wichtigen Netzwerkes ist. Durch eine Kompromittierung dieses Hosts wären indirekt auch die Dienste, die auf den Hosts im selben Netzwerk laufen, gefährdet.

Werden sowohl die IDS-Alarme als auch die aus den Host-Informationen gewonnenen Daten mit denen aus den Schwachstellendatenbanken kombiniert, ist es so möglich, die Priorisierung der einzelnen Sicherheitslücken zu verbessern. Die IDS-Alarme liefern hierbei die Aussage, wie aktiv eine Schwachstelle ausgenutzt wird, wohingegen die Host-Information eine Aussage darüber zulässt, wie verbreitet eine Schwachstelle und somit auch wie gefährlich sie für ein gegebenes System ist.

Durch die Verknüpfung der Netzwerktopologie und der Host-Information ist es im

Falle eines für einen Host eingehenden Alarmes möglich, zu bestimmen, wie viele andere Schwachstellen in Reichweite des betroffenen Hosts sind, deren Ausnutzung zur Kompromittierung wichtiger Ziele führen könnte. Dadurch würde sich eine bessere Priorisierung des Alarms erstellen lassen, da sich so die Gefährdung des gesamten Netzwerks im Falle einer erfolgreichen Übername des Hosts besser einschätzen ließe.

Werden die Flow-Informationen mit denen der Netzwerktopologie kombiniert, ist es möglich, die momentane Auslastung des Netzwerks zu bestimmen. Dies kann dazu verwendet werden, um Alarme, die Komponenten betreffen, welche ohnehin schon an der Grenze ihrer Belastbarkeit sind, höher zu bewerten, da ein Angriff auf diese Komponenten mit einer höheren Wahrscheinlichkeit zu einem Ausfall führen kann.

Wenn man die Flow-Daten unter Verwendung der Host-Informationen mit den Informationen der Dienststruktur verbindet, ist es möglich, anhand des im Netzwerk aufgezeichneten Datenverkehrs die Priorität der Dienste zu verbessern.

Die in der Schwachstellendatenbank enthaltenen Informationen über Sicherheitslücken können durch die Informationen über die Interna der Hosts auf die dort vorhandene Software abgebildet werden. Dadurch ist es möglich, für den Host einen Grad der aktuellen Gefährdung zu bestimmen. Außerdem können die so gewonnenen Informationen über die Anfälligkeit einer Anwendung in Bezug auf Vertraulichkeit, Integrität und Verfügbarkeit wiederum durch die Verbindung zwischen den Hosts und der Dienststruktur auf Dienste abgebildet werden. Daraus lässt sich die Gefährdung für einzelne Dienste bestimmen, welche dann für die Priorisierung von Alarmen bei einem Angriff auf einen Dienst verwendet werden kann.

Werden die Flow-Daten sowie die Informationen der Schwachstellendatenbanken über die Hosts auf die bereitgestellten Dienste abgebildet, können Informationen darüber gewonnen werden, wie stark ein Dienst frequentiert wird. Wird ein Dienst zum Zeitpunkt eines Angriffes stark frequentiert, so kann der Angriff als wichtiger eingestuft werden als der selbe Angriff zu einem Zeitpunkt in dem der Dienst weniger stark frequentiert ist.

Da IDS-Alarme die Hosts als Quelle bzw. Ziel angeben, kann über sie der Sicherheitstatus der Hosts angenähert werden. Das Auftauchen eines internen Hosts als Quelle eines Angriffs ist signifikant höher zu bewerten als das Auftreten als Ziel eines Angriffs. Durch die Kombination der Host-Informationen mit den Diensten kann wiederum die Gefährdung der Dienste abgeschätzt werden.

3.8 Zusammenfassung

Um auf die zu Beginn der Arbeit gestellten wissenschaftlichen Fragen Antworten zu finden, wurden einige Informationsquellen analysiert. Es wurden daraus einige ausge-

wählt und vorgestellt, die sich für die Priorisierung von Sicherheitsalarmen eignen. Des Weiteren wurden Vorschläge ausgearbeitet, wie diese in der Priorisierung verwendet werden können (F.1). Parallel zu der Beschreibung der Quellen sind bestehende Möglichkeiten betrachtet worden, wie die Informationen aus diesen Quellen automatisiert erfasst werden können, soweit dies überhaupt möglich ist (F.3). Nach der Erläuterung der Informationsquellen wurden einige Kombinationsmöglichkeiten aufgestellt, die aufzeigen, wie diese Quellen verbunden werden können, um eine bessere Priorisierung von Alarmen zu ermöglichen (F.2).

Kapitel 4

Ansatz der Priorisierung

Nachdem in den vorangegangenen Kapiteln die verwandten Ansätze und die zur Verfügung stehenden Informationsquellen vorgestellt wurden, wird im folgenden Kapitel ein Ansatz ausgearbeitet, wie die Informationsquellen zusammengeführt und für die Priorisierung nutzbar gemacht werden können.

4.1 Beispiel-Netzwerk

Zur Veranschaulichung der vorgeschlagenen Berechnungsvorschriften und des Aufbaus des Priorisierungsansatzes wird das folgende, stark vereinfachte Beispiel-Netzwerk verwendet. 4.1 stellt hierbei die Netzwerktopologie mit zusätzlichen Informationen über die Schwachstellen auf den gegebenen Systemen dar. Die Grafik 4.1 zeigt die vereinfachte Darstellung der vorhandenen Layer-3-Netzwerke und des von ihnen unterstützten Routings, also den Zugriff von einem Netzwerk auf ein anderes. Hierbei ist zu beachten, dass zu jedem Layer-3-Netzwerk auch genau ein Layer-2-Netzwerk gehört.

Die gesamte Infrastruktur besteht somit aus den beiden internen Netzwerken Workspace und DMZ (Demilitarized Zone). Innerhalb des Workspace-Netzwerkes befinden sich vier Desktop-Systeme, die von den Mitarbeitern des Unternehmens verwendet werden, wohingegen sich innerhalb des DMZ-Netzwerkes die beiden Server befinden, die sowohl interne als auch externe Dienste erbringen. Über eine Firewall ist es externen Systemen möglich, die öffentlichen Dienste der DMZ zu nutzen. In umgekehrter Richtung ist es sowohl der DMZ als auch den Workspace-Systemen möglich, auf das Internet zuzugreifen. Ein zentraler Router ist für die Vermittlung zwischen den Netzwerken verantwortlich und jeweils ein Layer-2-Switch regelt die Kommunikation innerhalb der Netzwerke. Die ungerichteten Kanten zwischen den Netzwerkkomponenten stellen hierbei die physikalischen Verbindungen dar. Eine gerichtete Kante mit der Überschrift FlowX bedeutet, dass eine Kommunikation ausgehend von einem der Hosts zu einem an-

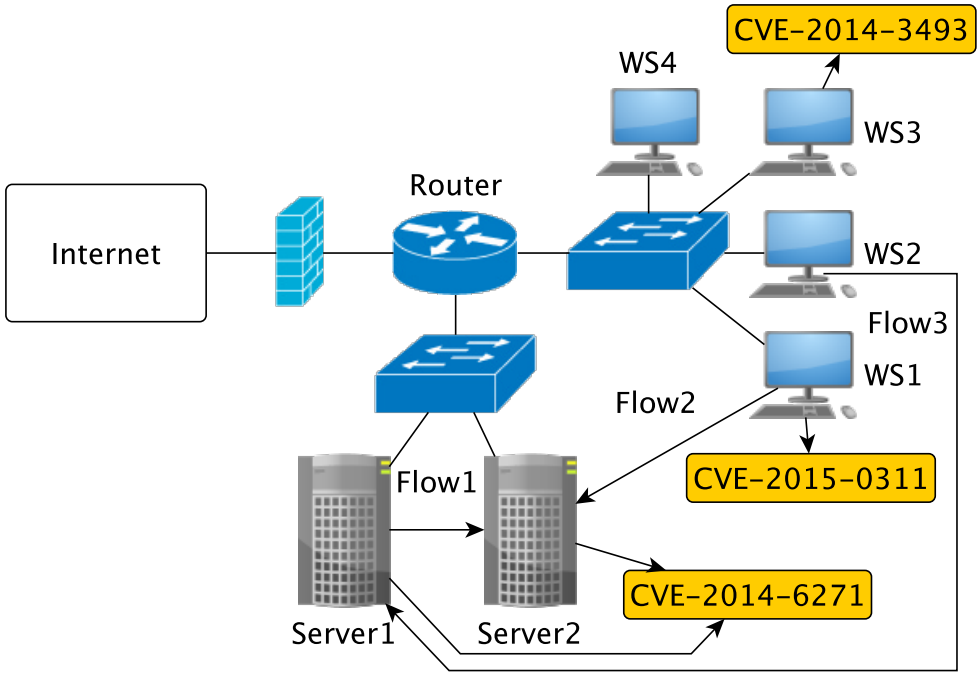


Abbildung 4.1: Hosts im Netzwerk

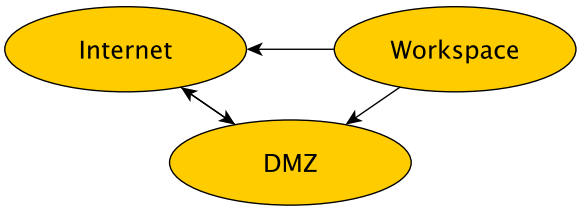


Abbildung 4.2: Routing

deren stattgefunden hat. Die Boxen mit CVE-Nummern beschreiben Sicherheitslücken, die Pfeile der Hosts auf sie die Verwundbarkeit eines Hosts in Bezug auf die gegebene Sicherheitslücke.

In Abbildung 4.3 sind die von den Servern der DMZ erbrachten Dienste und ihre Zusammenhänge dargestellt. Auf der untersten Ebene ist die Abbildung der Dienste auf die Server dargestellt. Die darüberliegende Ebene beschreibt die elementaren Dienste, aus denen über mehrere Ebenen hinweg sich der Gesamtdienst des Unternehmen zusammensetzt.

In den folgenden Abschnitten werden weitere Informationen über das Beispiel-Netz

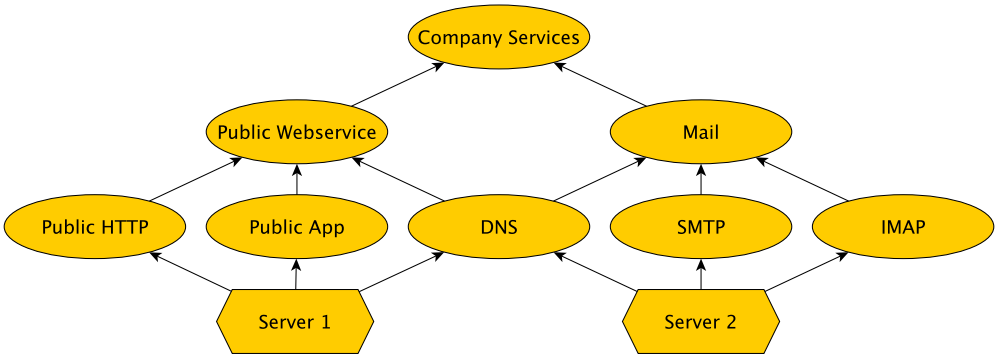


Abbildung 4.3: Dienststruktur & Server

bereitgestellt.

4.2 Grundlage: Graph

Am Beispiel ist erkennbar, dass die meisten Inhalte der Informationsquellen der Struktur eines Graphen sehr ähnlich sind und untereinander direkte Verbindungen aufweisen. Daher ist es naheliegend, auch die Zusammenführung der Informationsquellen mittels eines Graphen zu realisieren.

Aus diesem Grund wird zur Berechnung der Priorität eines Alarmes ein gerichteter Graph $G = (V, E, t_v, t_e)$ definiert. Dabei stellt V die Menge aller Knoten, E die Menge aller Kanten dar. Die Funktionen t_v und t_e sind dabei Abbildungsfunktionen der Knoten bzw. der Kanten auf definierte Mengen. Die genaue Definition der Abbildungsfunktionen ist in den Gleichungen 4.1 und 4.2 dargestellt.

$$t_v(x) = \begin{cases} \text{Service} & \text{wenn der Knoten einen Dienst abbildet} \\ \text{Device} & \text{wenn der Knoten ein Gerät / einen Host abbildet} \\ \text{Network} & \text{wenn der Knoten ein Netzwerk abbildet} \\ \text{Vulnerability} & \text{wenn der Knoten eine Schwachstelle abbildet} \\ \text{User} & \text{wenn der Knoten einen Nutzer darstellt} \end{cases} , x \in V \tag{4.1}$$

$$
\mathrm{t_e}(x) = \begin{cases}
\text{Flow} & \text{wenn die Kante einen Flow zwischen Hosts abbildet} \\
\text{PartOf} & \text{wenn die Kante die Bereitstellung einer Funktionalität zwischen Diensten abbildet} \\
\text{Provides} & \text{beschreibt die Bereitstellung eines Dienstes durch ein Device} \\
\text{AccessTo} & \text{wenn die Kante die Verbindung zwischen Host und Netzwerken beschreibt} \\
\text{VulnerableTo} & \text{wenn die Kante das Vorhandensein einer Verwundbarkeit zwischen einem Host und einer Schwachstelle beschreibt} \\
\text{Uses} & \text{bildet die Nutzung eines Dienstes durch einen Benutzer (User) ab}
\end{cases}, x \in E \tag{4.2}
$$

Mit Hilfe der Abbildungsfunktion des Graphen lassen sich schnittfreie Teilmengen der Knoten bzw. Kanten herleiten. Diese werden im Folgenden bei der Beschreibung einzelner Schritte der Berechnung der Priorität verwendet. Gleichung 4.3 stellt die Teilmengen der Knoten, Gleichung 4.4 die der Kanten dar.

$$
\begin{aligned}
\mathrm{V_S} &= \{v | \forall v \in V \wedge \mathrm{t_v}(v) = \text{Service}\} \\
\mathrm{V_D} &= \{v | \forall v \in V \wedge \mathrm{t_v}(v) = \text{Device}\} \\
\mathrm{V_N} &= \{v | \forall v \in V \wedge \mathrm{t_v}(v) = \text{Network}\} \\
\mathrm{V_V} &= \{v | \forall v \in V \wedge \mathrm{t_v}(v) = \text{Vulnerability}\} \\
\mathrm{V_U} &= \{v | \forall v \in V \wedge \mathrm{t_v}(v) = \text{User}\}
\end{aligned} \tag{4.3}
$$

$$
\begin{aligned}
\mathrm{E_F} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{Flow}\} \\
\mathrm{E_O} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{PartOf}\} \\
\mathrm{E_P} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{Provides}\} \\
\mathrm{E_C} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{ConnectedTo}\} \\
\mathrm{E_A} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{AccessTo}\} \\
\mathrm{E_V} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{VulnerableTo}\} \\
\mathrm{E_U} &= \{e | \forall e \in E \wedge \mathrm{t_e}(e) = \text{Uses}\}
\end{aligned} \tag{4.4}
$$

Da es um einen gerichteten Graphen handelt, gilt für die Kanten (a, b) und (b, a) wobei $a, b \in V$, dass $(a, b) \neq (b, a)$ auch wenn $\mathrm{t_e}((a, b)) = \mathrm{t_e}((b, a))$, da der erste Knoten in dem Kanten-Tupel den Anfang und der zweite das Ende einer Kante angibt.

4.3 Eingabewerte-Berechnung

Wird ein Alarm von einem IDS-System erzeugt und zur Priorisierung weitergeleitet, wäre eine mögliche Umsetzung, die Informationen direkt aus den verschiedenen Quellen zu holen und zu verbinden. Das Ad-hoc-holen und die im Graphen notwendigen Berechnungen für die Priorisierung wären jedoch sehr ressourcenaufwändig. Außerdem änderten sich einige der Informationen der Datenquellen zwischen zwei Alarmen nicht. Um die unnötige doppelte Berechnung der Werte zu vermeiden und die Priorisierung ressourcenschonender bzw. performanter zu machen, wurde das Problem der Berechnung der Priorität in zwei Schritte unterteilt. Die erste Hälfte des Algorithmus befasst sich hierbei mit der Berechnung der Eingabewerten für die Alarmpriorisierung, sozusagen mit den Vorberechnungen, die unabhängig von den eintreffenden Alarmen sind. Die zweite Hälfte, welche sich mit den Berechnungen pro eintreffendem Alarm befasst, wird in 4.4 beschrieben.

Da für die Priorisierung die Wichtigkeit des Hosts und der von ihm im umlegenden Netzwerk verursachbare Schaden als Eingabewerte von Interesse sind, werden diese vorberechnet. Dafür muss der Administrator zunächst für jeden Dienst den Wert des Dienstes $\text{value}_{\text{admin}}$ spezifizieren. Dies kann auf mehreren Stufen der Dienste-Hierarchie geschehen, es muss jedoch darauf geachtet werden, dass ein Wert nicht doppelt in die Berechnung einfließt. Außerdem muss der Administrator zusätzlich die Werte für $\text{confidentiality}_{\text{admin}}$. $\text{integrity}_{\text{admin}}$ und $\text{availability}_{\text{admin}}$ auf der untersten Ebene der Dienste spezifizieren. Die vier Werte werden dann über die Struktur der Dienste abgebildet, um den Wert eines Dienstes untergliedert in Vertraulichkeit ($\text{value}_{\text{conf}}$), Integrität ($\text{value}_{\text{integ}}$) und Verfügbarkeit ($\text{value}_{\text{avail}}$) zu erhalten. Damit die in dieser Arbeit vorgeschlagenen Formeln sinnvolle Ergebnisse liefern, ist zu beachten, dass alle vom Administrator eingegebenen Werte positiv sind.

Um bei der Abbildung von der Dienst-Hierarchie auf die Prioritäten der Alarme möglichst viele unterschiedliche Prioritätswerte zu erhalten, ist es ratsam, Werte ungleich 0 zu verwenden. Beispielsweise würde ein $\text{confidentiality}_{\text{admin}}$ Wert von 0 bedeuten, dass die unterschiedlichen Confidentiality-Impact-Werte der Sicherheitslücke keinen Unterschied im Ergebnis der Priorität mehr verursachen würden. Diese Information würde somit bei der Berechnung verloren gehen.

Im Beispiel wurden die folgenden Werte für $\text{value}_{\text{admin}}$ verwendet:

Dienstname	$\text{value}_{\text{admin}}$ in k $k = 1000\$$
Company Service	12
Public Webservice	10
Mail	11

Tabelle 4.1: Angegebene Werte der Dienste

Die $\text{confidentiality}_{\text{admin}}$. $\text{integrity}_{\text{admin}}$ und $\text{availability}_{\text{admin}}$ sind wie folgt:

Dienstname	$\text{confidentiality}_{\text{admin}}$	$\text{integrity}_{\text{admin}}$	$\text{availability}_{\text{admin}}$
Public HTTP	7,00	11,00	5,00
Public App	23,00	10,00	5,00
DNS	1,00	7,00	7,00
SMTP	13,00	10,00	5,00
IMAP	10,00	7,00	5,00

Tabelle 4.2: Angegebene CIA-Prioritäten der Dienste

Gleichung 4.5 beschreibt, wie die Confidentiality-, Integrity- und Availability-Werte (im Folgenden CIA-Werte) von der untersten Service-Ebene bis zum Service-Wurzelknoten abgebildet werden. Es handelt sich bei dieser Funktion um eine rekursive Definition. Das bedeutet, um den confidentiality-Wert eines Knoten zu berechnen, müssen vorher alle confidentiality-Werte der Knoten berechnet sein, die mittels einer PartOf-Relation auf den aktuellen Knoten verweisen. Das x in der Definition von N bezieht sich auf den Knoten, für den der confidentiality-Wert bestimmt werden soll. N ist hierbei so definiert, dass es jeweils die Kindelemente des aktuell ausgewählten Knotens enthält. Hierbei ist außerdem zu beachten, dass falls kein $\text{confidentiality}_{\text{admin}}$-Wert für einen Knoten spezifiziert wurde, dieser mit dem Wert 0 in die Berechnung einfließt. Analog dazu werden die Gleichungen für integrity und availability definiert.

$$\begin{aligned} \text{confidentiality}(x) = \text{confidentiality}_{\text{admin}}(x) + \sum_{j \in N} \text{confidentiality}(j), \\ N = \{(s | \forall s \in VS \rightarrow \exists (s,x) \in \text{E}_{\text{O}}\} \end{aligned} \tag{4.5}$$

Wird diese Formel auf die Werte des Beispiel-Netzwerkes angewendet, ergibt sich die Tabelle 4.3.

Dienstname	confidentiality	integrity	availability
Public Webservice	31,00	28,00	17,00
Mail	24,00	24,00	17,00
Company Services	55,00	52,00	34,00

Tabelle 4.3: Berechnete CIA-Werte, hochgerechnet

Nachdem die CIA-Werte aller Dienstknoten bestimmt wurden, werden diese nun verwendet, um die vom Administrator spezifizierten $\text{value}_{\text{admin}}$ Werte auf die unterste Dienstebene abzubilden. Somit werden für jeden Dienst die Werte für $\text{value}_{\text{conf}}$, $\text{value}_{\text{integ}}$ und $\text{value}_{\text{avail}}$ mittels der Gleichungen, die analog zu 4.7 aufgebaut sind, bestimmt. Die Einheit der Werte hängt von der Einheit ab, die für $\text{value}_{\text{admin}}$ verwendet wurde. Tabelle 4.4 enthält die für das Beispiel berechneten Werte.

$$\text{ciasum}(x) = \text{confidentiality}(x) + \text{integrity}(x) + \text{availability}(x) \tag{4.6}$$

$$\text{value}_{\text{conf}}(x) = \text{confidentiality}(x) \cdot \left(\frac{\text{value}_{\text{admin}}(x)}{\text{ciasum}(x)} + \sum_{j \in N} \frac{\text{value}_{\text{conf}}(j)}{\text{confidentiality}(j)} \right), \quad N = \{(s | \forall s \in \text{V}_\text{S} \rightarrow \exists (x, s) \in \text{E}_\text{O}\} \tag{4.7}$$

Dienstname	$\text{value}_{\text{conf}}$ in k	$\text{value}_{\text{integ}}$ in k	$\text{value}_{\text{avail}}$ in k
Company Services	4,68	4,43	2,89
Public Webservice	6,72	6,07	3,68
Mail	6,10	6,10	4,32
Public HTTP	1,52	2,38	1,08
Public App	4,98	2,17	1,08
DNS	0,47	3,30	3,30
SMTP	3,31	2,54	1,27
IMAP	2,54	1,78	1,27

Tabelle 4.4: Berechnete Prioritäten der Dienste

Ein Nachteil dieses Ansatzes ist, dass die Änderung eines $\text{confidentiality}_{\text{admin}}$-, $\text{integrity}_{\text{admin}}$- bzw. $\text{availability}_{\text{admin}}$-Wertes sehr kostspielig ist, da der gesamte Service & Device Baum neu berechnet werden muss. Dies gilt auch für das Hinzufügen bzw. Entfernen eines Dienstes. Dieser Fall tritt in einem echten Netzwerk jedoch nur mit großen Zeitabständen auf, da die Lebenszyklen der Dienste meist sehr groß sind.

Der Vorteil ist, dass die Werte einfacher zu bestimmen sind. Der Wert eines Dienstes kann über Einnahmen bzw. über etwaige Vertragsstrafen bestimmt werden. Die $\text{confidentiality}_{\text{admin}}$, $\text{integrity}_{\text{admin}}$ bzw. $\text{availability}_{\text{admin}}$ können sich direkt an den von den Diensten verwendeten Daten orientieren. Bei einer Änderung des $\text{value}_{\text{admin}}$ muss nur der jeweilige Teilbaum unterhalb des Dienstes neu berechnet werden.

Ein alternativer bzw. ergänzender Ansatz zur Berechnung von $\text{value}_{\text{conf}}$, $\text{value}_{\text{integ}}$ und $\text{value}_{\text{avail}}$ über die Dienste-Hierarchie ist, sie direkt über die vom Dienst verwalteten Daten zu bestimmen. Beispielsweise könnte bei einem Dienst, der Kundendaten vorhält, die Höhe der Entschädigung, die pro Kunde fällig wird, dafür verwendet werden. α beschreibt die Kosten pro Nutzer, die im Falle des Verlustes der Vertraulichkeit anfallen, in einer monetären Einheit (z. B. $).

$$\begin{aligned} \text{value}_{\text{conf}} &= \alpha \cdot |U|, \\ s &\in \text{V}_\text{S} \\ U &= \{u | \forall u \in \text{V}_\text{U} \rightarrow \exists (u,s) \in \text{E}_\text{U}\} \end{aligned} \tag{4.8}$$

Dies kann insbesondere dann von Interesse sein, wenn es durch spezielle Software (wie beispielsweise *SELinux*) auf den Systemen möglich ist, das Maß der auf einem Host vorhandenen Vertraulichkeit (durch Anzahl und Klassifizierung der Dokumente) zu bestimmen.

In jedem Fall wird $\text{value}_{\text{conf}}(s)$ von den Diensten auf $\text{value}_{\text{conf}}(d)$ der Devices, also auf die den Dienst bereitstellenden Hosts abgebildet, um es in die Priorisierung einfließen zu lassen. Gleichung 4.9 stellt die Berechnung von $\text{value}_{\text{conf}}(d)$ dar, die Werte von $\text{value}_{\text{integ}}(d)$ und $\text{value}_{\text{avail}}(d)$ werden analog berechnet.

$$\begin{aligned} \text{value}_{\text{conf}}(d) &= \sum_{j \in N} \frac{\text{value}_{\text{conf}}(j)}{|M|}, \\ d &\in \text{V}_\text{D} \\ N &= \{s | \forall s \in \text{V}_\text{S} \rightarrow \exists (d,s) \in \text{E}_\text{P}\} \\ M &= \{e | \forall e \in \text{V}_\text{D} \rightarrow \exists (e,j) \in \text{E}_\text{P}\} \end{aligned} \tag{4.9}$$

Aus der Anwendung von Gleichung 4.9 ergibt sich für das Beispiel die Tabelle 4.5.

Hostname	$\text{value}_{\text{conf}}$ in k	$\text{value}_{\text{integ}}$ in k	$\text{value}_{\text{avail}}$ in k
Server1	6,74	6,20	3,82
Server2	6,09	5,97	4,19
WS1, WS2, WS3, WS4	0,00	0,00	0,00

Tabelle 4.5: Berechnete CIA-Werte der Hosts

Nachdem die Dienststruktur verwendet wurde, um die Werte der Hosts basierend auf den bereitgestellten Diensten zu bestimmen, werden als Nächstes die Informationen der Schwachstellendatenbanken und der Netzwerktopologie in die Berechnung miteinbezogen. Das Ziel ist hierbei, eine Aussage darüber treffen zu können, wie viel Schaden ein Angreifer nach der Übernahme eines Hosts potentiell in dem umgebenden Netzwerk anrichten kann. Hierfür wird die Gleichung 4.10 aufgestellt. Die Abbildung 4.3 gibt einen Überblick mit Hilfe welcher weiterer Teilgleichungen der Wert von 4.10 berechnet wird. Zum besseren Verständnis der Zusammenhänge zwischen den Gleichungen werden diese hierarchisch strukturiert dargestellt.

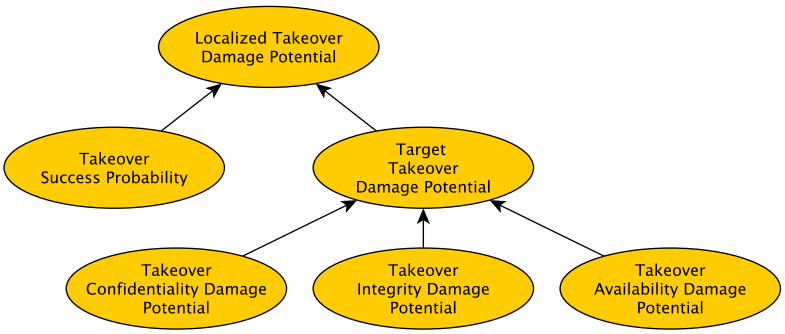


Abbildung 4.4: Gefährdung der Werte im Netzwerk ausgehend von einem Quellsystem

$$\mathrm{dam}_{\mathrm{ltf}}(d) = \sum_{i \in \mathrm{V_D}} \mathrm{p}_{\mathrm{fs}}(d,i) \cdot \mathrm{dam}_{\mathrm{tf}}(i) \tag{4.10}$$

Das Resultat von $\mathrm{dam}_{\mathrm{ltf}}$ (Localized Takeover Damage Potential) beschreibt, wie hoch der erwartete Schaden ist, falls Host d kompromittiert wird und der Angreifer sich diesen Umstand zunutzemacht, um weitere Systeme im Netz anzugreifen. Um diesen Wert zu bestimmen, wird ausgehend von dem Host d für jeden anderen Zielhost i der erwartete Schaden auf dem Host ($\mathrm{dam}_{\mathrm{tf}}$, Target Takeover Damage Potential) mit der Wahrscheinlichkeit, das dieser erfolgreich von dem Quellhost d angegriffen wird, kombiniert und die Ergebnisse dieser Kombinationen aufsummiert. Die Wahrscheinlichkeit p_{fs} (Takeover Success Probability) beschreibt somit, wie wahrscheinlich eine Ausbreitung des Angriffs von d in Richtung des Hosts i ist. Die Gleichung 4.11 zeigt auf, dass diese Wahrscheinlichkeit auf der Basis der Kommunikationsmöglichkeiten ($\mathrm{p}_{\mathrm{com}}$) zwischen a und b und der Wahrscheinlichkeit (p_{av}), dass das Zielsystem zum Zeitpunkt des Angriffs verfügbar ist, berechnet wird. Die Verfügbarkeit der Systeme spielt nur dann eine Rolle, wenn es sich nicht um zwei Server-Systeme sondern um eine Kombination aus Desktop und Serversystem handelt, bei der das Desktop-System regelmäßig abgeschaltet wird. Es wird also angenommen, dass das zu schützende Netzwerk nicht nur aus Server-Systemen, die eine Verfügbarkeit nahe zu 100% aufweisen, besteht. Gleichung 4.12 gibt an, wie diese Wahrscheinlichkeit aus den Messungen der Up- und Downtimes, in denen das System verfügbar bzw. abgeschaltet war, errechnet werden kann.

$$\mathrm{p}_{\mathrm{fs}}(a,b) = \mathrm{p}_{\mathrm{com}}(a,b) \cdot \mathrm{p}_{\mathrm{av}}(b) \tag{4.11}$$

$$\mathrm{p}_{\mathrm{av}}(b) = \frac{\mathrm{t}_{\mathrm{up}}(b)}{\mathrm{t}_{\mathrm{up}}(b) + \mathrm{t}_{\mathrm{down}}(b)} \tag{4.12}$$

$$p_{com}(a,b) = \begin{cases} 1 & \text{wenn eine direkte Kommunikation von Host a zu b stattgefunden hat.} \\ 0,8 & \text{wenn sich die Hosts im selben Layer-2-Netzwerk befinden.} \\ 0,3 & \text{wenn Host b über ein Layer-3-Netzwerk von Host a aus erreichbar ist.} \\ 0 & \text{sonst.} \end{cases} \tag{4.13}$$

Mit p_{com} wird beschrieben, wie wahrscheinlich die Verbreitung eines Angriffs von a in Richtung des Zielhosts b ist, gemessen an der Konnektivität der beiden Hosts. Somit werden sowohl die Flow- als auch die Netzwerktopologie-Informationen eingebunden. Die Werte orientieren sich hierbei an der Wahrscheinlichkeit der Ausbreitung eines Angriffs innerhalb eines Netzwerkes und beachten deshalb die Sichtbarkeit zwischen dem Quell- und dem Zielhost. Bei den Werten handelt es sich nur um einen initialen Vorschlag. Dieser sollte ggf. an die Gegebenheiten des zu schützenden Netzwerkes angepasst werden.

Das Ergebnis von $dam_{tf}(i)$, welches in Gleichung 4.10 verwendet wird, ist eine Kombination aus dem auf dem Zielhost i vorhandenen Wert und einer Abschätzung, wie stark dieser durch vorhandene Sicherheitslücken gefährdet ist. Dies ist in Gleichung 4.14 beschrieben und kann im unteren Abschnitt der Abbildung 4.10 nachvollzogen werden. Somit setzt sich dam_{tf} für ein gegebenes System aus den Werten für Vertraulichkeit (dam_{tc}, Takeover Confidentiality Damage Potential), Integrität (dam_{ti}, Takeover Integrity Damage Potential) und Verfügbarkeit (dam_{ta}, Takeover Availability Damage Potential) zusammen. Da somit alle auf dem Host vorhandenen Werte einbezogen werden und nicht nur die des angegriffenen Dienstes, wird eine Abschätzung getroffen, die den schlimmsten Fall betrifft.

$$dam_{tf}(d) = dam_{tc}(d) + dam_{ti}(d) + dam_{ta}(d) \tag{4.14}$$

Da die Terme für dam_{tc}, dam_{ti} und dam_{ta} gleich aufgebaut sind, wird hier nur auf dam_{tc} näher eingegangen. Die anderen Funktionen sind analog dazu definiert. Der Wert für dam_{tc} setzt sich somit nach Formel 4.15 aus der maximalen, negativen Auswirkung (maximalen imp_c) auf die Vertraulichkeit, die eine auf dem System vorhandene Sicherheitslücke zulässt, multipliziert mit dem monetären Wert ($value_{conf}$) der vertraulichen Informationen, die auf dem Host vorhanden sind, zusammen.

$$dam_{tc}(b) = imp_{mc}(b) \cdot value_{conf}(b) \tag{4.15}$$

$$\text{imp}_{\text{mc}}(b) = \begin{cases} \max_{v \in N} \text{imp}_{\text{c}}(v) & \text{wenn } |N| > 0 \\ \alpha & \text{sonst.} \end{cases} \tag{4.16}$$
$$N = \{v | \forall v \in \text{V}_\text{V} \rightarrow \exists (b, v) \in \text{E}_\text{V}\}$$

imp_{mc} in Gleichung 4.15 beschreibt die größte Bedrohung der Vertraulichkeit, die auf dem Host vorhanden ist. Gleichung 4.16 beschreibt, wie dieser Wert aus den auf dem Host vorhandenen Sicherheitslücken bestimmt wird. Für den Fall, dass keine Sicherheitslücke auf dem Host vorhanden ist, wird der Minimalwert α angenommen, da eine Sicherheit von 100% nicht wahrscheinlich ist. Die Hilfsmenge N beinhaltet hierbei alle Sicherheitslücken, die auf dem gegebenen System b vorhanden sind.

Die Berechnung von imp_{c} basiert hierbei auf den durch die Schwachstellendatenbank gewonnenen Informationen über die negative Auswirkung der Sicherheitslücke auf die Vertraulichkeit des Systems.

$$\text{imp}_{\text{c}}(v) = \begin{cases} \text{cvss}_{\text{ConfidentialityImpact}}(v) & \text{wenn } \exists\, \text{cvss}_{\text{ConfidentialityImpact}}(v) \\ \max \text{cvss}_{\text{ConfidentialityImpact}} & \text{sonst.} \end{cases} \tag{4.17}$$

$$\text{cvss}_{\text{ConfidentialityImpact}}(v) = \begin{cases} 0,0 & \text{wenn ConfidentialityImpact} = \text{none} \\ 0,275 & \text{wenn ConfidentialityImpact} = \text{partial} \\ 0,660 & \text{wenn ConfidentialityImpact} = \text{complete} \end{cases} \tag{4.18}$$

Die Definitionen von $\text{cvss}_{\text{IntegrityImpact}}$ und $\text{cvss}_{\text{AvailabilityImpact}}$ sind gleich wie die von $\text{cvss}_{\text{ConfidentialityImpact}}$ und basieren auf [14].

Nachdem alle Formeln für die Berechnung der Eingabewerte aufgestellt sind, können diese dazu verwendet werden, die Berechnung der Eingabedaten für das Beispiel durchzuführen.

Für diese Berechnung werden neben den Eingabewerten des letzten Abschnitts auch noch die Informationen der Sicherheitslücken benötigt, die in in den Tabellen 4.6 und 4.7 gegeben sind und auf den Daten von [36] basieren.

Vulnerability	$\text{cvss}_{\text{ConfidentialityImpact}}$	$\text{cvss}_{\text{IntegrityImpact}}$	$\text{cvss}_{\text{AvailabilityImpact}}$
CVE-2014-6271	0,66	0,66	0,66
CVE-2015-0311	0,66	0,66	0,66
CVE-2014-3493	0	0	0,27

Tabelle 4.6: Sicherheitslücken: Beeinträchtigung aus Schwachstellendatenbank

Vulnerability	$\text{cvss}_{\text{AccessComplexity}}$	$\text{cvss}_{\text{AccessVector}}$	p_{vc}
CVE-2014-6271	0,71	Network	1
CVE-2015-0311	0,71	Network	1
CVE-2014-3493	0,71	Adjacent	0

Tabelle 4.7: Sicherheitslücken: Erreichbarkeit aus Schwachstellendatenbank

Hostname	imp_{mc}	imp_{mi}	imp_{ma}
Server1, Server2	0,66	0,66	0,66
WS1, WS2, WS4	0,1	0,1	0,1
WS3	0,1	0,1	0,275

Tabelle 4.8: Durch Schwachstellen auf Hosts berechnete Impact-Werte

Tabelle 4.8 stellt hierbei die maximalen Einfluss-Werte der vorhandenen Sicherheitslücken, die mittels der Formeln 4.16, 4.17 und 4.18 sowie den äquivalent aufgebauten Formeln für Integrität und Verfügbarkeit berechnet wurden, dar.

Werden alle über das Netzwerk vorhandenen Informationen zusammengeführt, wird die Tabelle 4.9 erzeugt. An dieser können beispielsweise die Auswirkungen der Eingabedaten der Diensthierarchie auf die bereitstellenden Systeme betrachtet werden. Server1 und Server2 heben sich, bezogen auf die Werte, die auf den Servern direkt vorhanden sind, also dam_{tc}, dam_{ti} und dam_{ta} von den Workstations ab. Die Auswirkung der Netzwerktopologie ist deutlich an den dam_{ltf}-Werten der Workstations zu sehen, da diese durch die dam_{ta}-Werte der Server erzeugt werden. Der prioritätssteigernde Effekt der Flow-Informationen ist deutlich an der Differenz der Prioritäten zwischen den WS1 und WS2 zu WS3 und WS4 zu erkennen. Die Tabelle 4.9 enthält somit die Eingabewerte, die im Folgenden mit weiteren Informationen nach Eintreffen eines Alarms kombiniert werden, um die Priorität des Alarms zu berechnen.

4.4 Prioritätshierarchie

Nachdem die vorbereitenden Berechnungen der Eingabewerte vorgestellt wurden, befasst sich der folgende Abschnitt mit den Berechnungen, die zur Erzeugung der Priorität nach Eintreffen eines Alarms verwendet werden.

$$\text{priority}(a) = \text{dam}_{\text{ltf}}(\text{source}(a)) + \text{dam}_{\text{hac}}(a) + \text{dam}_{\text{etd}}(a), \tag{4.19}$$

Das Ziel der Priorisierung ist, wie bereits in 2.8 beschrieben, einem Alarm einen numerischen Wert zuzuweisen, der dazu genutzt werden kann, die Wichtigkeit dieses Alarms mit derjenigen anderer Alarme zu vergleichen. Die oberste Stufe der Hierarchie stellt

Hostname	dam_{tc}	dam_{ti}	dam_{ta}	dam_{tf}	dam_{ltf}
Server1	4,45	4,09	2,52	11,06	10,72
Server2	4,02	3,94	2,77	10,72	8,84
WS1	0,00	0,00	0,00	0,00	14,04
WS2	0,00	0,00	0,00	0,00	14,27
WS3	0,00	0,00	0,00	0,00	6,53
WS4	0,00	0,00	0,00	0,00	6,53

Tabelle 4.9: Berechnete Hostübernahme-Prioritäten

Abbildung 4.5: Erste Ebene der Prioritätshierarchie

deshalb die Priorität dar, welche den Schaden, der durch einen Angriff im Unternehmen angerichtet werden könnte, beschreibt. Wie in Abbildung 4.4 bzw. in Gleichung 4.19 dargestellt, setzt sich diese aus den drei Eingabewerten: $dam_{ltf}(\text{source}(a))$ (Source Localized Takeover Damage Potential), dam_{hac} (Expected Takeover Damage Potential) und dam_{etd} (Expected Direct Damage Potential) zusammen. Der Term $dam_{ltf}(\text{source}(a))$ umfasst hierbei die Bedrohung, die für das Netzwerk entsteht, wenn die Quelle des Angriffs bzw. des Alarmes a, $source(a)$ sich im Inneren des zu schützenden Netzwerks befindet. Der Term ist natürlich nur dann vorhanden, wenn kein Zweifel an der Herkunft des Angreifers besteht. Insbesondere wird er bei Angriffen, bei denen ein Spoofing, also das Fälschen der Quell-IP-Adresse möglich ist, nicht verwendet. Dies bedeutet, der Wert des Terms wird als 0 angenommen. dam_{hac} beschreibt den Schaden für das gesamte Netzwerk außer dem Zielsystem, der erwartet wird, falls das System erfolgreich von einem Angreifer übernommen wird. Zu guter Letzt beziffert dam_{etd} die Kosten, die direkt auf dem angegriffenen System entstehen, falls der Angriff erfolgreich war.

Nachdem die Beschreibung der Teilhierarchie unterhalb der Definition von $dam_{ltf}(\text{source}(a))$ bereits vorgestellt wurde, folgt nun die Beschreibung des Terms dam_{hac} aus Abbildung 4.19. In Abbildung 4.4 bzw. in Gleichung 4.20 ist die Zusammensetzung von dam_{hac} dargestellt. Dieser Term besteht aus der Wahrscheinlichkeit, dass eine Übernahme (p_{hac}) des Zielsystems $\text{target}(a)$ durch den aktuell zu bewertenden Angriff erfolgt, multipliziert mit dem erwarteten Schaden (dam_{ltf}, siehe 4.10), der durch die Übernahme des Hosts im Netzwerk eintreten könnte.

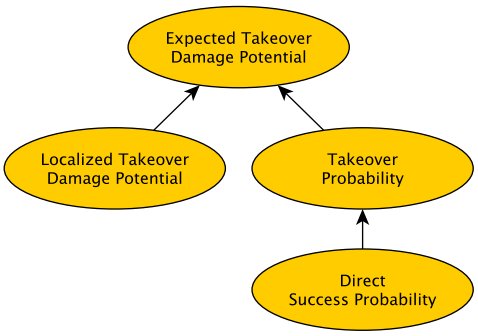


Abbildung 4.6: Abschätzung der mittelbaren Bedrohung durch eine Übernahme des Zielsystems

$$\mathrm{dam}_{\mathrm{hac}}(a) = \mathrm{p}_{\mathrm{hac}}(a) \cdot \mathrm{dam}_{\mathrm{ltf}}(\mathrm{target}(a)) \tag{4.20}$$

Die Wahrscheinlichkeit für $\mathrm{p}_{\mathrm{hac}}$ setzt sich zusammen aus der Erfolgswahrscheinlichkeit des Angriffs (p_{d}) sowie der Wahrscheinlichkeit (p_{vc}), dass es dem Angreifer nach der erfolgreichen Ausführung des Angriffs möglich ist, weitere Angriffe von dem Zielsystem aus auf andere Systeme des Netzwerks zu starten. Dies würde bedeuten, dass es dem Angreifer möglich geworden ist, die Kontrolle über das angegriffene System zu übernehmen (Gleichung 4.21).

$$\mathrm{p}_{\mathrm{hac}}(a) = \mathrm{p}_{\mathrm{vc}}(a) \cdot \mathrm{p}_{\mathrm{d}}(a) \tag{4.21}$$

Der Wert der p_{d} des Alarms lässt sich dadurch ermitteln, dass die Werte für p_{vs}, p_{ve} sowie p_{ha} multipliziert werden (Gleichung 4.22). Die Hilfsfunktion vulnerability(a) ermittelt hierbei aus den Informationen des Alarms a die verwendete Schwachstelle v.

$$\begin{aligned} \mathrm{p}_{\mathrm{d}}(a) &= \mathrm{p}_{\mathrm{vs}}(v) \cdot \mathrm{p}_{\mathrm{ve}}(t,v) \cdot \mathrm{p}_{\mathrm{ha}}(t), \\ v &= \mathrm{vulnerability}(a), \\ t &= \mathrm{target}(a) \end{aligned} \tag{4.22}$$

p_{vs} beschreibt, wie wahrscheinlich eine erfolgreiche Ausnutzung der Sicherheitslücke ist. Beispielsweise reicht bei Angriffen, die auf der Ausnutzung des Laufzeitverhaltens (Race Conditions) der Zielanwendung beruhen, ein einzelner Angriff oft nicht aus, um die Sicherheitslücke erfolgreich auszunutzen. Im Gegensatz zu p_{vs} wird durch p_{ve} be-

schrieben, wie wahrscheinlich es ist, dass die Sicherheitslücke auf dem Host vorhanden ist und zum Zeitpunkt des Angriffs ausgenutzt werden konnte. p_{ha} schließlich gibt an, wie wahrscheinlich es ist, dass das angegriffene Zielsystem im Augenblick des Angriffes erreichbar war. Dies lässt sich, wie beispielsweise in 4.23 dargestellt, berechnen. Hierbei beschreibt der Term $\text{duration}_{\text{fail}}(d)$ die Zeit, die seit dem Erkennen des Ausfalls des Hosts d in Sekunden vergangen ist. Die Variable y kann dazu verwendet werden, um die Dauer eines Ausfalls anzugeben, ab der ein System als dauerhaft ausgefallen betrachtet wird und somit die Wahrscheinlichkeit für die Wiederherstellung der Verfügbarkeit bzw. Angreifbarkeit auf den Basiswert von 0,1 sinkt. Die Gleichung ist so gestaltet, dass die Wahrscheinlichkeit für die Angreifbarkeit des Hosts im Falle eines Ausfalls mit der Dauer des Ausfalls abnimmt.

$$p_{ha}(d) = \begin{cases} 1 & \text{wenn der Host aktiv ist} \\ (1 - \left(\frac{\text{duration}_{\text{fail}}(d)}{y}\right)^2) \cdot 0,9 + 0,1 & \text{wenn } \text{duration}_{\text{fail}}(d) \leq y \\ 0,1 & \text{sonst} \end{cases} \tag{4.23}$$

In Gleichung 4.24 wird dargestellt, wie der Wert der p_{ve} errechnet wird. Falls es dem System möglich war, zu ermitteln, dass der gegebene Host für die Vulnerability v anfällig ist, wird die Wahrscheinlichkeit als 100% angenommen. Ist dies nicht der Fall, so wird für die Bestimmung der Wahrscheinlichkeit der $p_{\text{SecurityScannerTrust}}(v)$ verwendet, also mit welcher Wahrscheinlichkeit der eingesetzte Vulnerability Scanner die gegebene Sicherheitslücke erkennt.

$$p_{ve}(d,v) = \begin{cases} 1 & \text{wenn } \exists (d,v) \in E_V \\ 1 - p_{\text{SecurityScannerTrust}}(v) & \text{sonst.} \end{cases} \tag{4.24}$$

Wie der Gleichung 4.25 zu entnehmen ist, wird für p_{vs} die $\text{cvss}_{\text{AccessComplexity}}$ aus der Schwachstellendatenbank verwendet. Die dafür möglichen Werte sind in 4.26 (nach [14]) beschrieben.

$$p_{vs}(a) = \begin{cases} \text{cvss}_{\text{AccessComplexity}}(v) & \text{wenn } v = \text{vulnerability}(a) \rightarrow \exists\, \text{cvss}_{\text{AccessComplexity}}(v) \\ 1 & \text{sonst.} \end{cases} \tag{4.25}$$

$$\text{cvss}_{\text{AccessComplexity}}(v) = \begin{cases} 0.35 & \text{wenn high} \\ 0.61 & \text{wenn medium} \\ 0.71 & \text{wenn low} \end{cases} \tag{4.26}$$

Da eine große Anzahl der Sicherheitslücken nicht für eine Übernahme eines Systems verwendet werden kann, wird p_{vc} (Gleichung 4.27) verwendet, sodass die dam_{hac} nur dann mit in die Priorität einfließt, wenn die aktuell angegriffene Sicherheitslücke die Übernahme des Systems ermöglicht.

$$p_{vc}(a) = \begin{cases} 1 & \text{wenn die Sicherheitslücke dem Angreifer eine} \\ & \text{Übernahme des Systems erlaubt.} \\ 0.1 & \text{sonst.} \end{cases} \tag{4.27}$$

Neben dem Einfluss von p_d auf dam_{hac} wird es auch verwendet, um dam_{etd} in der Gleichung 4.19 zu bestimmen. Die Verwendung wird in Abbildung 4.7 bzw. Gleichung 4.28 dargestellt. Die unmittelbare Bedrohung (dam_{etd}, Expected Direct Damage Potential) durch das dem Alarm zugrunde liegende Ereignis setzt sich zusammen aus der Wahrscheinlichkeit (p_d, Direct Success Probability), dass der Angriff erfolgreich war und dem maximalen Schaden (dam_d, Direct Damage Potential), der unmittelbar auf dem Host durch den gegebenen Angriff entstanden ist.

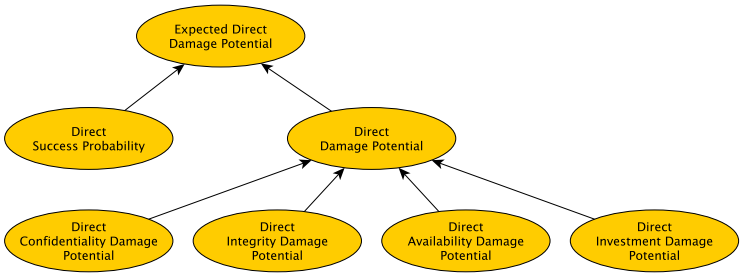


Abbildung 4.7: Abbildung der unmittelbaren Bedrohung

$$dam_{etd}(a) = p_d(a) \cdot dam_d(a) \tag{4.28}$$

Analog zu der Berechnung von dam_{tf} (4.14) setzt sich die Berechnung von dam_d aus der Summe der erwarteten Verluste an Vertraulichkeit, Integrität und Verfügbarkeit zusammen.

Der Unterschied der Gleichungen liegt darin, dass im Falle von dam_{dc}, dam_{di} und dam_{da}, wie in 4.30 dargestellt, die Informationen der angegriffenen Sicherheitslücke und nicht der maximal negative Einfluss der auf dem Host vorhandenen Sicherheitslücken zum Einsatz kommt.

$$dam_d(a) = dam_{dc}(a) + dam_{di}(a) + dam_{da}(a) \tag{4.29}$$

$$\mathrm{dam}_{\mathrm{dc}}(a) = \mathrm{imp}_{\mathrm{c}}(\mathrm{vulnerability}(a)) \cdot \mathrm{value}_{\mathrm{conf}}(\mathrm{target}(a)) \tag{4.30}$$

Abschliessend werden die soeben vorgestellten Formeln anhand des Beispiel-Netzwerkes und einiger Testalarme demonstriert.

Simulierter Angriff	$\mathrm{dam}_{\mathrm{ltf}}$	$\mathrm{dam}_{\mathrm{hac}}$	$\mathrm{dam}_{\mathrm{etd}}$	priority in k	priority in $
1. CVE-2015-0311 auf WS4	0,00	0,93	0,00	0,93	927,83
2. CVE-2015-0311 auf WS1	0,00	9,97	0,00	9,97	9969,31
3. CVE-2014-6271 auf Server1 mit Fix	0,00	1,52	1,57	3,09	3092,76
4. CVE-2014-6271 auf Server1	0,00	7,61	7,85	15,46	15463,80
5. CVE-2014-3493 auf Server1	0,00	1,52	0,15	1,67	1669,04
6. CVE-2014-3493 von WS4 auf Server1	6,53	1,52	0,15	8,20	8203,04

Tabelle 4.10: Berechnete Alarm-Prioritäten

Die Ergebnisse dieser simulierten Testangriffe können an der Tabelle 4.10 nachvollzogen werden. Der erste Angriff stellt die Priorität eines nicht zentralen Hosts dar, auf den ein Angriff ausgeführt wird, für den er nicht verwundbar ist. Wohingegen der zweite Angriff einen aufgrund dieser Sicherheitslücke verwundbaren Host trifft, der auch noch bereits Verbindungen zu einem der Server aufgebaut hatte. Dies wird auch in dem deutlichen Unterschied zwischen den Prioritäten der Angriffe eins und zwei sichtbar. In den Prioritäten zu Angriff drei und vier ist der Unterschied zwischen Werten bei dem Vorhandensein einer Sicherheitslücke auf einem Host deutlich zu sehen. Für Angriff drei wurde deshalb abweichend zur Definition des Beispiel-Netzwerks angenommen, dass die Sicherheitslücke „CVE-2014-6271“ nicht mehr auf dem Server angreifbar ist. Die letzten beiden Angriffe stellen den Effekt dar, der entsteht, wenn ein Angriff von außerhalb bzw. von innerhalb des Netzwerkes erfolgt. Da beim zweiten Angriff der Wert für $\mathrm{dam}_{\mathrm{ltf}}$ der Workstation 4 mit in die Priorität einfließt, ist ein starker Anstieg der Priorität zu sehen.

4.5 Zusammenfassung

In diesem Kapitel wurde die Berechnungsvorschrift des in dieser Arbeit ausgearbeiteten Priorisierungsansatzes systematisch dargestellt und mit einem Beispiel begleitend erläutert. Der Berechnungsprozess wurde dabei in zwei Phasen aufgeteilt. Im ersten Schritt wurden die Vorberechnungen der Eingabedaten aufgezeigt. Dafür wurde zu Beginn die Datenstruktur beschrieben, die dazu verwendet werden kann, die Daten der Informationsquellen zusammenzuführen. Die so errechneten Eingabewerte fließen

im zweiten Schritt in die Berechnung der Priorität der Alarme ein. Die Berechnung der Priorität wurde hierbei über eine Hierarchie von Gleichungen dargestellt. In einem abschließenden Beispiel wurden die Formeln exemplarisch für die Berechnung verschiedener Angriffsszenarien verwendet.

Kapitel 5

Design

Zur Evaluierung des ausgearbeiteten Priorisierungsansatzes soll ein Prototyp entwickelt werden, dessen Struktur und Funktion im folgenden Kapitel näher erläutert wird.

5.1 Komponenten

Die in dieser Arbeit entwickelten Module zur Priorisierung von Alarmen sind als Teil der Software *VITSI (Verteilte IT Sicherheitsinfrastruktur)* entworfen worden. *VITSI* bietet die Möglichkeit, die Priorisierung direkt in den Prozess des Empfangens eines Alarms umzusetzen. Für diese Arbeit wurde jedoch entschieden, die Priorisierung in einem nachgelagerten eigenständigen Modul durchzuführen. Die für diese Arbeit wichtigen Komponenten des *VITSI*-Systems sind in Abbildung 5.1 dargestellt.

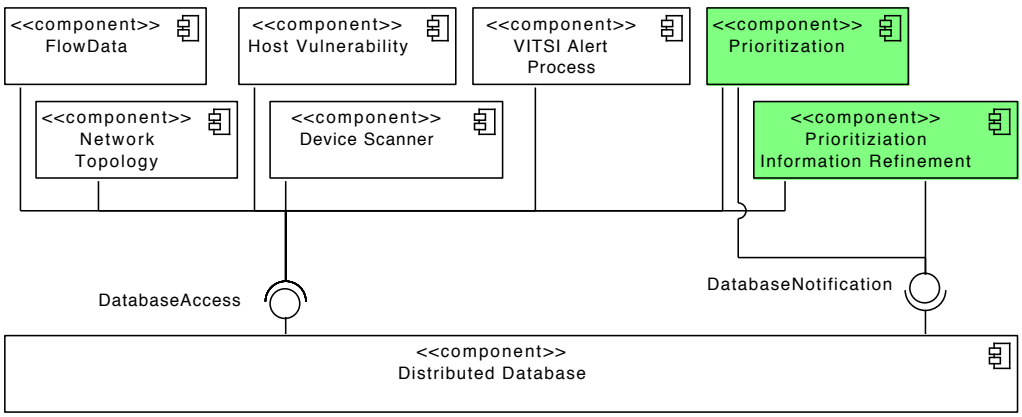


Abbildung 5.1: *VITSI* Module

Die Umsetzung der eigentlichen Priorisierung erfolgt in zwei getrennten Modulen. Diese sind in der Abbildung farblich markiert. Hierbei ist das Modul *Prioritization*

Information Refinement dafür verantwortlich, die von anderen Modulen bereitgestellten Informationen zu verbinden und für die Prioritätsberechnung vorzubereiten. Das Modul *Prioritization* führt auf diesen Informationen aufbauend die eigentliche Priorisierung der einzelnen Alarme aus.

Wie in der Abbildung 5.1 zu sehen ist, verfügt jede der für die Priorisierung verwendeten Datenquellen über ein eigenes Modul innerhalb des *VITSI*-Systems. Diese Module sind dafür verantwortlich, die extern gesammelten Daten an die interne Datenstruktur anzupassen und in der Datenbank zu speichern. Die darunterliegende verteilte Datenbank ist die Basis von *VITSI* und zuständig für die Ausfallsicherheit und Synchronisation der Daten zwischen den Systemen. Des Weiteren gibt es für die Module die Möglichkeit, sich für Benachrichtigungen über Änderungen der Datensätze in der Datenbank bzw. dem Graphen zu registrieren. Die Kommunikation zwischen den Modulen basiert vollständig auf der Datenbank. Dies bedeutet, es gibt keine direkte Kommunikation zwischen den einzelnen Modulen.

5.2 Datenmodell

Das Datenmodell der in der Datenbank gespeicherten Daten ist an den bereits in Kapitel 4.2 vorgestellten Graphen angelehnt. Es ist jedoch geringfügig detaillierter als der Graph, da es die reale Struktur der Systeme möglichst genau abbilden soll. Das gesamte Datenmodell und die Verbindungen sind in Abbildung 5.2 als UML-Klassen-Diagram dargestellt. Das Modell beschränkt sich auf die Klassen und deren Verbindungen. Die Attribute sind hierbei nicht dargestellt. Im Folgenden werden die Klassen anhand ihrer Zugehörigkeit zu den einzelnen in Kapitel 3 vorgestellten Datenquellen erklärt.

Die Informationen über die Netzwerktopologie werden über die Klassen **Network**, **Layer2Network**, **Layer3Network** und **IpAddress** abgebildet. Die Klasse **Network** modelliert ein Netzwerk im Allgemeinen. Aus ihr werden die speziellen Formen **Layer2Network** und **Layer3Network** abgeleitet. Durch die generalisierte Verbindung **AccessTo** auf **Network** ist es möglich, sowohl den Zugriff auf Layer-3-Netzwerke über Layer-2-Netzwerke als auch das Routing zwischen Layer-3-Netzwerken darzustellen.

Bei der Untersuchung eines Layer-3-Netzwerkes ist es möglich, die in ihm vorhandenen IP-Adressen zu erkennen. Diese werden auf die Klasse **IpAddressen** abgebildet. Sie stellen den Schnittpunkt zwischen den Informationen der Netzwerktopologie über Layer-3-Netzwerke und die Host-Informationen dar. Durch das Untersuchen der Hosts ist es möglich, die durch den Netzwerk-Scan gefundenen IP-Addressen auf die Hosts abzubilden.

Weitere wichtige Informationen sind hierbei die auf den Hosts vorhandenen Netzwerk-Interfaces (**NetworkInterface**) und MAC-Adressen (**MacAddress**), welche die Verbin-

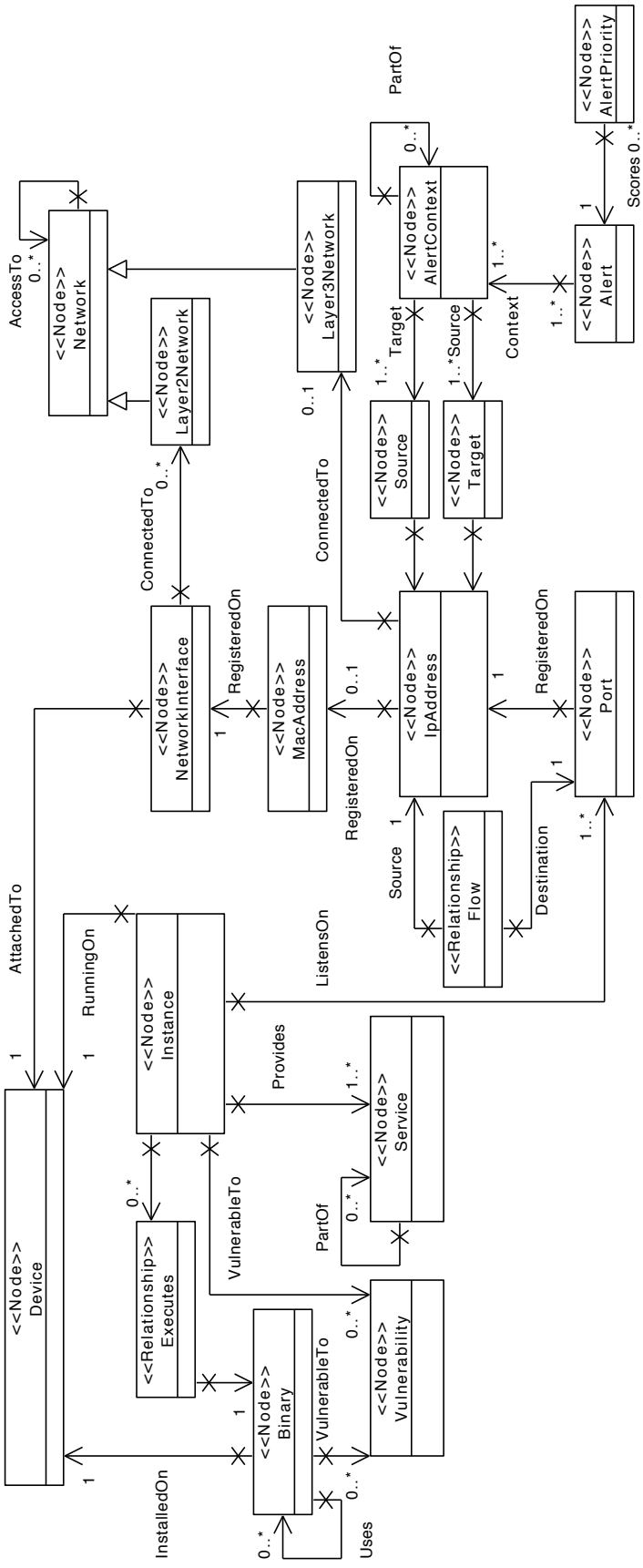


Abbildung 5.2: Datenmodell

dung zu den **Layer2Networks** der Netzwerktopologie darstellen. Die Hosts werden durch die Klasse **Device** abgebildet. Die Klasse **Instance** stellt hierbei einen konfigurierten Dienst dar und bildet den Schnittpunkt zur Diensthierarchie. Die Dienste und ihre hierarchische Abhängigkeiten werden durch die Klasse **Service** und die dazugehörige **PartOf** Kante abgebildet. Von einer Instanz gestartete Prozesse werden über die Klasse **Executes** dargestellt, welche wiederum die Binaries (**Binary**), also Programme und Bibliotheken des Hosts nutzen. Die auf den Hosts gesammelten Informationen über Schwachstellen in den Anwendungen können, je nachdem wie detailliert die zur Verfügung stehende Information ist, über die Kante **VulnerableTo** zwischen der Klasse **Vulnerability** und **Instance** oder **Binary** dargestellt werden.

Die Klasse **Flow** bildet den Fluss von Informationen zwischen einer Quell-IP-Adresse und einem Ziel-Port ab. Sie zeigt somit, dass ein Datenfluss von einem Quellhost zu einem Zielhost stattgefunden hat.

Schließlich werden die Informationen der Alarme über die Klassen **Alert** und **AlertContext** gespeichert. **AlertContext** stellt über **Source** und **Target** die Verbindung zur Netzwerktopologie durch die Informationen der Quelle und des Ziels des Angriffs her. Über die Kante **Scores** wird die erzeugte Priorität dem dazugehörigen Alarm zugewiesen.

5.3 Prozess der Priorisierung

Im vorangegangenen Abschnitt wurden die Struktur der Module und das Datenmodell, über das sie kommunizieren, vorgestellt. Es folgt nun die Beschreibung des Prozesses, auf dem die Priorisierung basiert. Abbildung 5.3 stellt den Gesamtprozess in Form eines mit BPMN (Business Process Model and Notation) erstellten Diagramms dar. Aus der Darstellung lässt sich leicht ablesen, dass die Erfassung der Informationsquellen durch die einzelnen Module parallel stattfinden kann. Für die Vorverarbeitung ist jedoch ein gewisses Mindestmaß an Informationen erforderlich. Für die Prüfung dieser Bedingung dient das komplexe Gateway nach den Prozessen für die Erfassung bzw. Aktualisierung der Informationsquellen. Diese Basisinformationen, die vor dem ersten Durchlauf der Vorverarbeitung in einer ersten Version vorhanden sein müssen, sind die Netzwerktopologie, die Dienststruktur einschließlich der vom Administrator eingegebenen Werte und die Abbildung der Dienste auf den Hosts. Nachdem dieser erste Durchlauf der Vorverarbeitung der Informationsquellen durchgeführt wurde, können die so erzeugten Informationen für die Priorisierung genutzt werden. Nach dem ersten Durchlauf ist es sinnvoll, dass die Daten der Informationsquellen aktualisiert bzw. erweitert werden, um so die Priorisierung zu verbessern. Damit die Änderungen der Daten der Informationsquellen in die errechneten Prioritäten einfließen, ist es allerdings notwendig, dass die Vorverarbeitung der Informationsquellen komplett oder zum Teil erneut ausgeführt

wird.

Die Priorisierung eines Alarms findet, wie im unteren Teil des Prozess-Diagramms dargestellt, angestoßen durch den Eingang der Benachrichtigung über einen neuen Alarm statt.

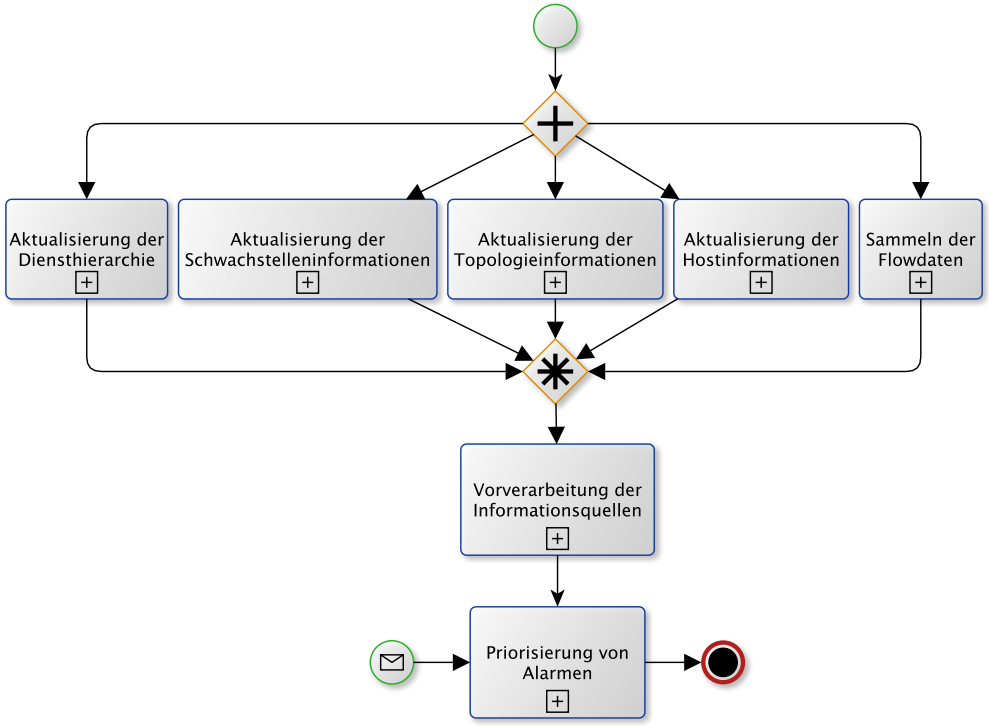
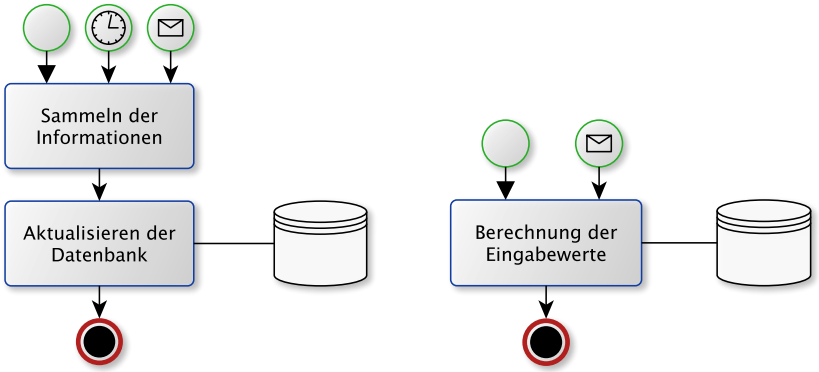


Abbildung 5.3: Gesamtprozess der Priorisierung

Abbildung 5.4(a) stellt die generelle Funktionsweise der Quellmodule dar. Hierbei kann das Sammeln der Informationen durch drei Ereignisse ausgelöst werden. Je nach Informationsquelle kann dies durch eine Zeitsteuerung, das manuelle Auslösen oder eine Benachrichtigung über eine Änderung in den Eingabedaten sein. Nach der Informationsbeschaffung folgt im nächsten Schritt die Aktualisierung der in der Datenbank vorhandenen Informationen. Dies kann das Löschen, Hinzufügen oder die Aktualisierung eines oder mehrerer Datensätze bedeuten. Durch das Durchführen dieser Aktualisierung der Daten kann wiederum der Prozess der Vorverarbeitung der so geänderten Eingabedaten erfolgen.

In Abbildung 5.4(b) wird der Prozess der Vorverarbeitung der Informationsquellen genauer dargestellt. Er wird manuellen oder durch eine Änderung in den Daten ausgelöst, wird hierbei der in 4.3 beschriebene Algorithmus zur Zusammenführung der Daten der Informationsquellen ganz oder nur zum Teil ausgeführt. Nach der erfolgreichen

Ausführung des Algorithmus stehen die von der Priorisierung benötigten Werte in den dazugehörigen Datenobjekten der Datenbank. Insbesondere werden die Werte für dam_{ltf}, $value_{conf}$, $value_{integ}$ und $value_{avail}$ in den dazugehörigen Attributen der Instanzen vom Typ **Device** des jeweiligen Hosts gespeichert.



(a) Sammeln der Informationen (b) Vorverarbeitung

Abbildung 5.4: Quellen & Refinement

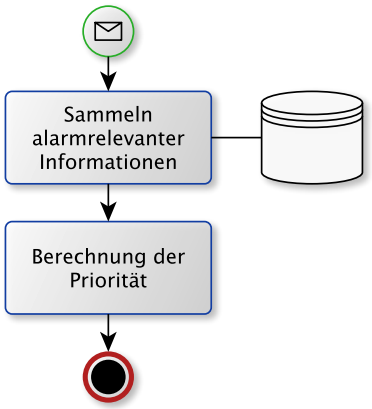


Abbildung 5.5: Berechnen der Priorität eines Alarms

In Abbildung 5.5 ist der Prozess der Berechnung detailliert dargestellt.

Für die Berechnung sind zwei Schritte erforderlich. Angestoßen durch den Eingang eines neuen Alarms, wird im ersten Schritt die Sammlung der für den gegebenen Alarm notwendigen Informationen aus der Datenbank durchgeführt. Die so gesammelten Werte werden dann mittels der in Abschnitt 4.4 beschriebenen Prioritätshierarchie zu einem Prioritätswert vereint.

5.4 Zusammenfassung

Um die Priorität eines Alarmes nach dem ausgearbeiteten Ansatz berechnen zu können, wurde die Grundlage für eine prototypische Umsetzung als Teil des *VITSI*-Systems ausgearbeitet. Hierfür wurde der in Kapitel 4.2 beschriebene Graph auf ein Datenmodel abgebildet und beschrieben, wie die Prozesse aufgebaut werden können, die für die Berechnung der Priorität notwendig sind.

Kapitel 6

Prototyp

Nachdem im vorangegangenen Kapitel die Rahmenbedingungen für eine Umsetzung des Priorisierungsansatzes beschrieben wurden, werden im folgenden Kapitel die wichtigsten Elemente der prototypischen Implementierung dargestellt. Zuerst werden die für den Prototypen ausgewählten Techniken dargestellt, gefolgt von einer Übersicht über die entwickelten Softwaremodule und deren Funktion.

6.1 Ausgewählte Techniken

Im Folgenden werden die Techniken behandelt, die für die Entwicklung des Prototypen ausgewählt wurden. Hierbei wird zunächst die jeweilige Technik beschrieben, gefolgt von einer Erklärung, warum sie für den Prototypen als geeignet befunden wurde. Zuletzt wird dargelegt, wie die genannte Technik eingesetzt wurde.

Zur Umsetzung des Prototypen wurde die Programmiersprache *Java* in der Version 8 ausgewählt. Die Auswahl der Programmiersprache *Java* basierte darauf, dass der Prototyp mit in das Ergebnis des ANSII-Projekts [37] integriert werden sollte, welches bereits in *Java* entwickelt wurde. Die Entscheidung für die Version 8 wurde getroffen, um von neuen Sprachkonzepten zu profitieren, insbesondere von der in *Java* 8 eingeführten *Stream-API*, den Lambda Ausdrücken und dem überarbeiteten Date and Time API [38]. Ein weiterer Grund waren die Vielzahl der vorhandenen Frameworks für *Java*, von denen einige im Folgenden näher vorgestellt werden.

Als Datenbank wurde die Graphdatenbank *Neo4j* ausgewählt [39]. Der wichtigste Grund für die Auswahl von *Neo4j* war die einfache Integrierbarkeit in den Prototypen, da beide in *Java* geschrieben sind. Außerdem bietet *Neo4j* in der Enterprise Edition eine Cluster-Lösung, welche unter den Bedingungen der *AGPL* (Affero General Public License) unentgeltlich verwendet werden kann [40]. Durch die sehr mächtige Abfragesprache Cypher in *Neo4j* wird es möglich, den gespeicherten Graphen nach Strukturmustern zu

untersuchen. *Neo4j* stellt somit die verwendete Clusterdatenbank, in der alle Informationen gespeichert werden, die für die Priorisierung benötigt werden.

Da *Neo4j* ohne Weiteres keine Möglichkeit bietet, dass sich eine Anwendungskomponente auf Änderungen des Graphen registriert, wurde am *Lehrstuhl für Netzarchitekturen und Netzdienste* das von der *Graph Aware Ltd.* [41] entwickelte *GraphAware ChangeFeed Module Neo4j*-Plugin als Basis für diese Funktion verwendet. Um eine gezieltere Registrierung auf Änderungen anbieten zu können, wurde es um die Funktionalität erweitert, die es ermöglicht, dass ein Client über Änderungen in Knoten mit bestimmten Labels benachrichtigt wird. Labels beziehen sich hierbei auf den Typ der Klasse. Es ist beispielsweise möglich, eine Benachrichtigung für alle Änderungen der Host-Informationen zu erhalten. Diese Funktionalität wird im Prototyp sowohl von dem Modul *Prioritization* für die Benachrichtigung über neu erstellte Alarme als auch für die Koordination zwischen dem Modul *Prioritization Information Refinement* und den Quellmodulen verwendet.

Das in 5.2 beschriebene Datenmodell wurde auf *Java*-Klassen abgebildet. Nachdem die Entscheidung für *Neo4j* getroffen war, musste ein Weg gefunden werden, mit möglichst geringem Aufwand die Verbindung zwischen den Modellklassen und dem Graphenmodell von *Neo4j* herzustellen. Für dieses Mapping wurde das Spring Data Framework *Neo4j* ausgewählt. Mit diesem wird ermöglicht, einen Großteil des *Java*-Sourcecodes, der sonst für die Abbildung der *Java*-Klassen auf die in *Neo4j* verwendeten Knoten- und Kantentypen dient, durch Annotationen zu ersetzen. Außerdem stellt es für einen Großteil der Datenbankoperationen und vor allem für die Suchmethoden auf den Attributen automatisch erzeugte Funktionen bereit. Nur bei sehr komplexen Abfragen ist das manuelle Schreiben von Cypher-Abfragen bzw. der Implementierungen notwendig. Deshalb wurden alle Klassen des Datenmodells mit den notwendigen Annotationen versehen und von Spring Data verwendete Repository Interfaces angelegt. [42]

JAXB (Java Architecture for XML Binding) ist der Standard dafür, wie *Java*-Klassenhierarchien auf XML-Dateien abgebildet werden können. Um die Klassen des Datenmodells nicht nur auf die Graphdatenbank abzubilden, sondern auch in der Lage zu sein, sie im XML-Format zu speichern, wurden sie zusätzlich mit *JAXB*-Annotationen versehen. [43]

CDI (Context Dependency Injection) ist ein Framework, das es *Java*-Entwicklern unter anderem ermöglicht, anstatt einer Verbindung zwischen expliziten Klassen auf Interfaces zu setzen. Die Interfaces werden dann über den Context, in dem die Anwendung gestartet wird, auf die konkreten Implementierungen abgebildet [44]. *CDI* wurde somit eingesetzt, um Interfaces von den konkreten Implementierungen zu entkoppeln. Die verwendete *CDI*-Implementierung war *Weld* [45].

JUnit ist ein Framework zur Erstellung von Testfällen für *Java*-Anwendungen. Durch die Implementierung von Testfällen ist es möglich, auch während des Entwicklungsprozesses sicherzustellen, dass ein Fehler im erwarteten Verhalten der Anwendung bereits

vor dem Testen in einer Test-/Produktivumgebung auffällt. Außerdem ermöglichen die Testfälle eine Konservierung der erzielten Ergebnisse. Es ist beispielsweise möglich, einen Performance-Test jederzeit ohne große Aufwände zu wiederholen. *JUnit* wurde insbesondere bei der Entwicklung des Testszenarios sowie in der Erstellung der *JAXB* und Spring-Data-*Neo4j*-Annotationen verwendet [46].

Die so erstellten Testfälle konnten des Weiteren durch die Verwendung von Maven als Build Tool automatisiert werden. Der Vorteil dieser Automatisierung liegt darin, dass der Entwickler nicht mehr an die Ausführung des jeweiligen Testfalles denken muss und frühzeitig auf Fehler hingewiesen wird, die sonst zu Problemen bei anderen Entwicklern führen können. Maven ermöglicht es, die verwendeten Bibliotheken getrennt von dem entwickelten Sourcecode zu verwalten und sorgt, richtig angewendet, für eine bessere Modularisierung der entwickelten Software.

Als einheitliche Group-ID für alle Module wurde hierbei *de.vitsi* gewählt. Bei allen Artefakten wurde die Group-ID als Prefix vor den eigentlichen Artefaktnamen gestellt, um die Lesbarkeit der erzeugten Jar-Namen zu verbessern. [47]

6.2 Module

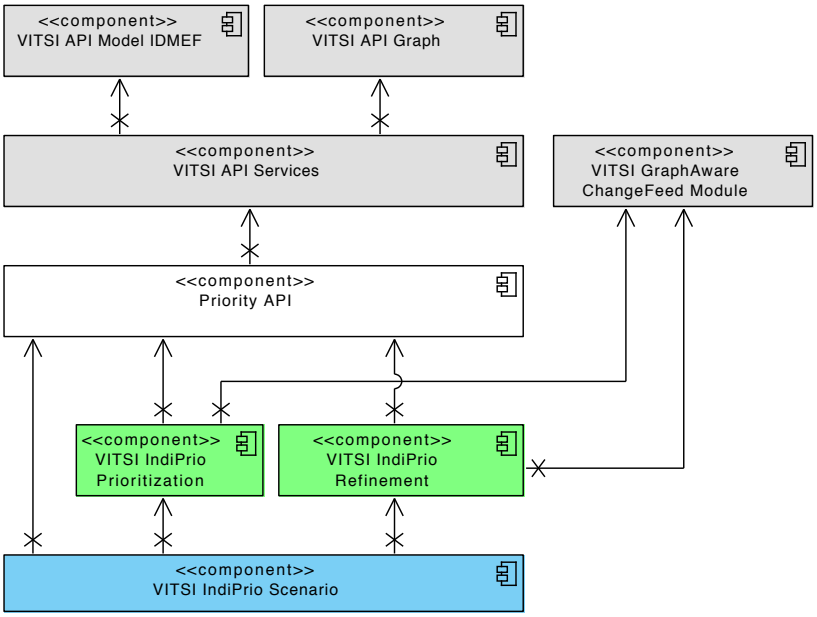


Abbildung 6.1: Module des Priorisierungsprototypen

Der für diese Arbeit entwickelte Prototyp für die Priorisierung befindet sich innerhalb des Maven-Modules mit der Artefakt ID: *de.vitsi.module.indiprio* als Teil der *VITSI*-Software. Im Diagramm 6.1 sind die Abhängigkeiten der einzelnen Untermodule

dargestellt.

Eine Kante bedeutet hierbei, dass ein Modul ein anderes benötigt und somit auch Zugriff auf die Klassen hat, die von diesem Modul bereitgestellt werden. Die grau markierten Module sind Teil des *VITSI*-Systems und somit nicht direkt ein Teil der Umsetzung des Priorisierungsansatzes. Das weiß markierte Modul enthält die Datenmodell-Klassen und -Interfaces, die für die Kommunikation zwischen den Implementierungs-Modulen verwendet werden. Die Implementierungs-Module, die einen Teil des in 4 beschriebenen Priorisierungsansatzes umsetzen, also die Implementierung enthalten, sind grün markiert. Blau markiert ist das Testmodul, das nur für die Evaluierung des Ansatzes entwickelt wurde und somit nicht für die Priorisierung notwendig ist.

6.2.1 Priorisierungs API

Das Priorisierungs-API-Modul (Artefakt ID: *de.vitsi.module.indiprio.api*) enthält sowohl die von den Priorisierungsmodulen verwendeten *Java*-Interfaces als auch das für ihre Kommunikation notwendige Datenmodell.

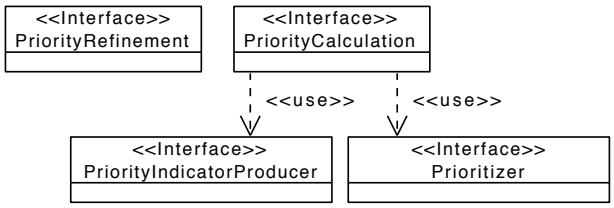


Abbildung 6.2: Interfaces der Priorisierung

Das Diagramm 6.2 stellt die bereitstehenden *Java*-Interfaces dar. **PriorityRefinement** führt die Eingabedaten der Informationsquellen innerhalb der Datenbank zusammen. Die Aufgabe der Priorisierung eines konkreten Alarms wird durch die Implementierung des **PriorityCalculation**-Interface durchgeführt. Die Implementierung dieser Schnittstelle kann hierbei auf die Interfaces **PriorityIndicatorProducer** und **Prioritizer** zurückgreifen. Der **PriorityIndicatorProducer** ist dafür zuständig, die Indikatoren, die für die Berechnung eines Alarms notwendig sind, aus der Datenbank zu holen. Die Implementierung des Prioritizer Interface setzt die Berechnung der Prioritätshierarchie um.

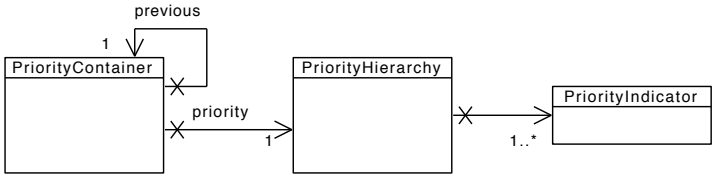


Abbildung 6.3: Prioritätshierarchie

Das Ergebnis der Prioritätsberechnung wird in einem Priority Container gespeichert. Dieser ist auch in der Lage, bei mehrfacher Berechnung der Priorität die vergangenen Ergebnisse zu erhalten. Die Zwischenergebnisse und die errechnete Priorität werden in einer **PriorityHierarchy** hinterlegt. Hier ist für jedes Zwischenergebnis der Prioritätsberechnung ein **PriorityIndicator** gespeichert. Die Zusammenhänge dieser Klassen werden in 6.3 dargestellt. Innerhalb der **PriorityHierarchy** gibt es ein Attribut *priority* welches den errechneten Prioritätswert inklusive der maximal und minimal möglichen Priorität enthält. Für alle Eingabedaten sowie die Zwischenergebnisse der Prioritätsberechnung werden somit **PriorityIndicator**-Instanzen erzeugt.

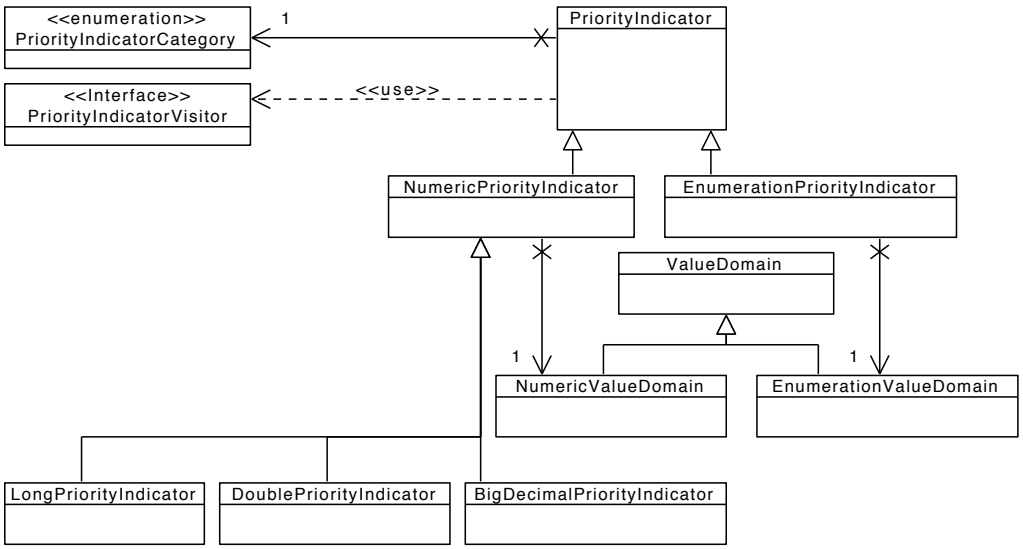


Abbildung 6.4: Priority-Indikatoren

Diese Indikatoren werden auch für den Fluss der Informationen vom **PriorityIndicatorProducer** bis zum **Prioritizer** verwendet. Das Datenmodell ist in Abbildung 6.4 dargestellt. Das wichtigste Merkmal eines Indikators ist hierbei die **PriorityIndicatorCategory**. Diese sorgt dafür, dass der gegebene Eingabewert an der richtigen Stelle der Prioritätsberechnung eingesetzt wird. Die Prioritätsberechnung wählt anhand dieses Merkmals die korrekte Formel für die Kombination mehrerer Indikatoren der jeweiligen Kategorie aus. Außerdem wird die **PriorityIndicatorCategory** dafür verwendet, um die korrekte Stelle auszuwählen, an der das Ergebnis der Berechnung innerhalb der **PriorityHierarchy** eingesetzt wird. Die **ValueDomain** enthält den minimal als auch maximal möglichen Prioritätswert, der aus der Berechnung resultieren könnte. Sie kann dazu verwendet werden, um eine bessere Einschätzung des resultierenden Prioritätswertes zu erhalten.

6.2.2 Refinement

Das Modul *VITSI IndiPrio Refinement* (Artefakt ID: *de.vitsi.module.indiprio.refinement*) stellt die Implementierung des Interface **PriorityRefinement** dar, welches eine Implementierung des in 4.3 vorgestellten Algorithmus zur Berechnung der für die Priorisierung verwendeten Eingabewerte beinhaltet. Das Modul verwendet hierzu die *Neo4j*-Datenbank, um die Prioritätswerte der Hosts sukzessive zu bestimmen und speichert auch die Resultate der Berechnung im Graphen ab.

6.2.3 Prioritization

Das Modul *VITSI IndiPrio Prioritization* (Artefakt ID: *de.vitsi.module.indiprio.prioritization*) implementiert die Interfaces **PriorityCalculation** und **Prioritizer** des API und setzt somit den in 4.4 vorgestellten Algorithmus zur Priorisierung von Alarmen um.

6.2.4 Szenario

Das Modul *VITSI IndiPrio Szenario* (Artefakt ID: *de.vitsi.module.indiprio.scenario*) prüft die Implementierung des Priorisierungs-Ansatzes anhand eines Beispiel-Netzwerkes. Es stellt eine erweiterte Version des bereits in 4.1 vorgestellten Netzwerkes dar. Die Ausführung eines Testdurchlaufs auf diesem Beispiel-Netzwerk erfolgt basierend auf einer temporären Datenbank und wird in drei Schritten ausgeführt:

1. Erzeugung des Netzwerkes und der benötigten Eingabedaten in der *Neo4j*-Datenbank.
2. Ausführung der Vorverarbeitung der Prioritäts-Informationen basierend auf den in der Datenbank vorhandenen Informationen.
3. Ausführung einiger beispielhafter Prioritätsberechnungen und Aufzeichnung der Dauer der Berechnungen.

Bei der Umsetzung des Szenarios wurde auf die Funktionalität des Modules *ChangeFeed* verzichtet, da der Fokus auf der Evaluierung der Implementierung des Priorisierungsansatzes und der Bewertung der Performance der Module ohne externe Einflüsse lag.

6.3 Zusammenfassung

Da die Entwicklung des Prototypen maßgeblich durch die verwendeten Techniken beeinflusst wurde, wurden diese zu Beginn des Kapitels vorgestellt. Anschließend wurden wichtige Informationen zu den einzelnen Modulen des Prototypen zusammenfassend dargestellt.

Kapitel 7

Evaluierung

Der im letzten Kapitel vorgestellte Prototyp sollte unter anderem dazu dienen, den ausgearbeiteten Priorisierungsansatz zu bewerten. Hierzu wird als Erstes das im Modul 6.2.4 entwickelte Szenario vorgestellt. Danach erfolgt eine, vom Prototypen unabhängige Analyse der Eigenschaften der Gleichung, die zur Berechnung der Prioritäten entwickelt wurde. Anschließend folgt ein kurzer Überblick über die Performance der Implementierung des Ansatzes, der mit Hilfe des Testszenarios und einiger Prioritätsberechnungen ermittelt wurde. Zum Schluss werden die Vor- und Nachteile des Ansatzes diskutiert und in einer Vergleichstabelle zusammengefasst.

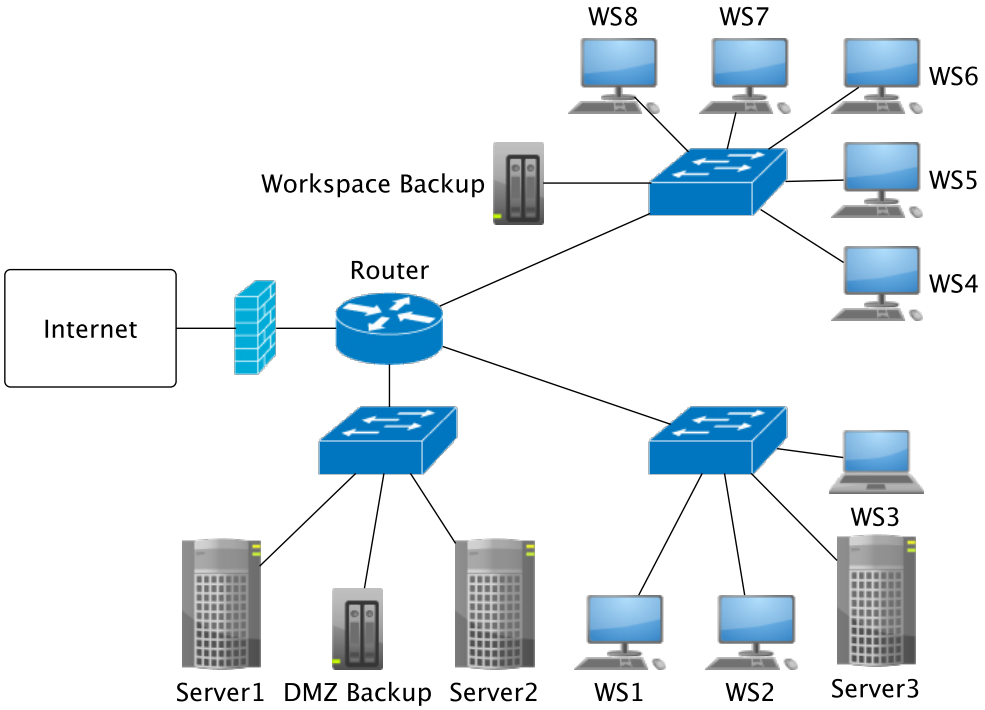


Abbildung 7.1: Szenario Netzwerk

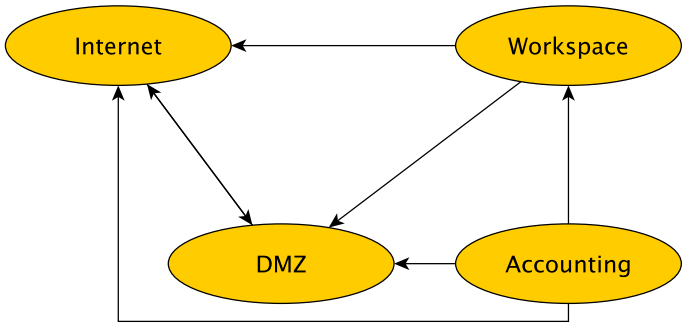


Abbildung 7.2: Szenario Routing

7.1 Szenario

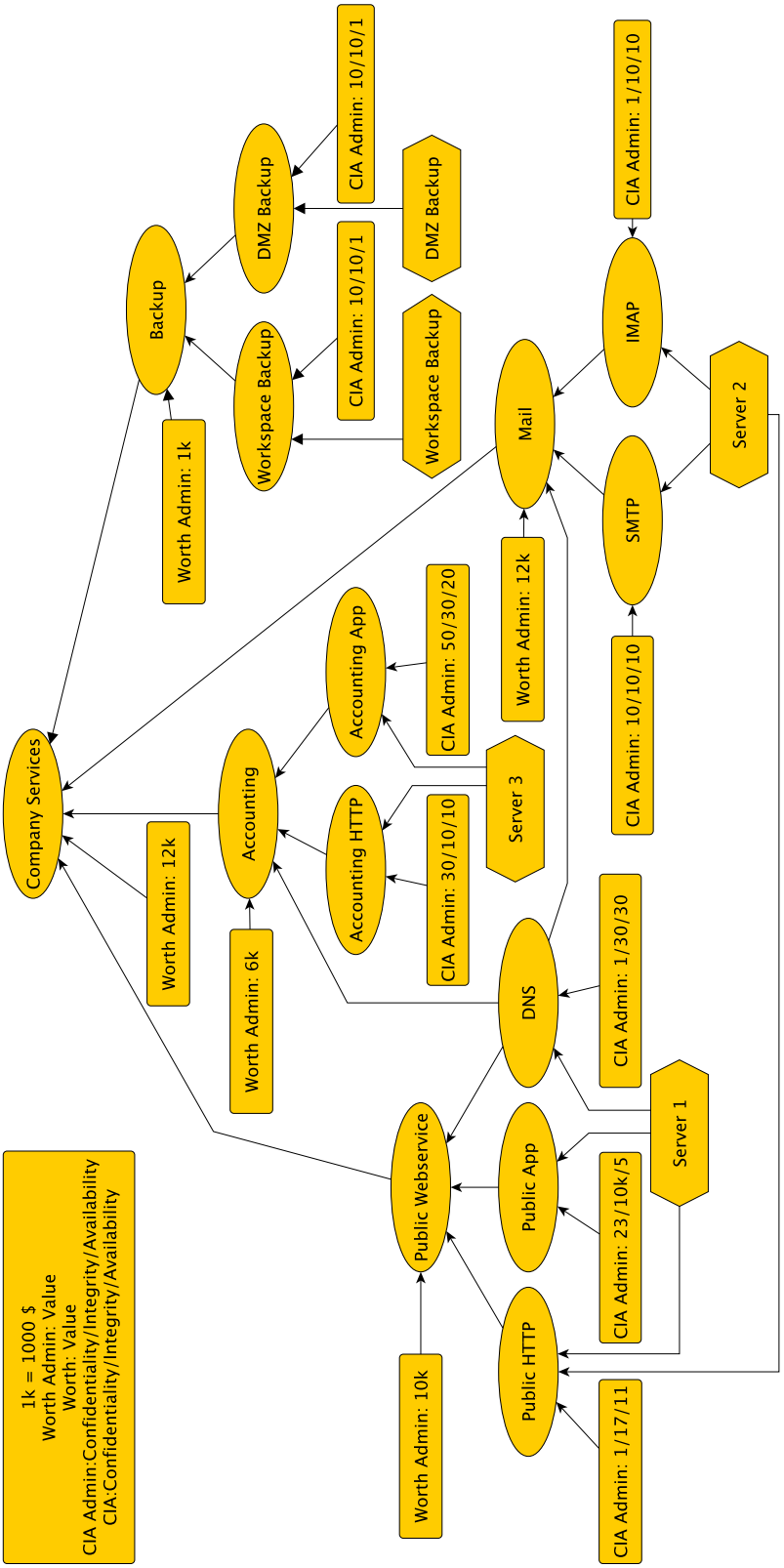


Abbildung 7.3: Szenario Dienststruktur

Vulnerability	$cvss_{ConfidentialityImpact}$	$cvss_{IntegrityImpact}$	$cvss_{AvailabilityImpact}$
CVE-2014-0001	0,275	0,275	0,275
CVE-2014-0004	0,66	0,66	0,66
CVE-2014-0018	0	0,275	0
CVE-2014-0019	0	0	0,275
CVE-2014-0020	0	0	0,275
CVE-2014-0254	0	0	0,66
CVE-2014-0319	0	0,66	0
CVE-2014-6278	0,66	0,66	0,66

Tabelle 7.1: Sicherheitslücken: Beeinträchtigung aus Schwachstellendatenbank im Szenario

Vulnerability	$cvss_{AccessComplexity}$	$cvss_{AccessVector}$	p_{vc}	Host
CVE-2014-0001	0,71	NETWORK	0,1	Server1
CVE-2014-0004	0,61	LOCAL	0,1	
CVE-2014-0018	0,61	LOCAL	0,1	
CVE-2014-0019	0,61	LOCAL	0,1	
CVE-2014-0020	0,71	NETWORK	0,1	
CVE-2014-0254	0,71	NETWORK	0,1	WS8
CVE-2014-0319	0,61	NETWORK	0,1	
CVE-2014-6278	0,71	NETWORK	1	Server1

Tabelle 7.2: Sicherheitslücken: Erreichbarkeit aus Schwachstellendatenbank im Szenario

In den Abbildungen 7.1, 7.2 und 7.3 sind die Netzwerktopologie, das Routing sowie die Dienststruktur des Szenario-Netzwerkes dargestellt, die an das Beispiel in 4.1 angelehnt sind. Das Szenario umfasst somit drei verschiedene Netzwerke und die externe Anbindung an das Internet. Das DMZ-Netzwerk (DMZ = Demilitarized Zone, deutsch: demilitarisierte Zone) ist, wie auch im ersten Beispiel, für die Bereitstellung der internen und externen Dienste verantwortlich. Das Workspace-Netzwerk umfasst PCs für die Mitarbeiter des Unternehmens. Hinzugekommen ist das extra abgesicherte Buchhaltungsnetzwerk (Accounting) und der dazugehörige Buchhaltungsserver. Das Buchhaltungsnetzwerk hat, wie in 7.2 dargestellt, sowohl Zugriff auf die DMZ als auch auf das Workspace-Netzwerk. Die Tabellen 7.1 und 7.2 enthalten die Informationen über die im Szenario vorhandenen Sicherheitslücken.

Im Zuge dessen wurde auch die Diensthierarchie um die Dienste für die Buchhaltung (Accounting) sowie einen Backup-Dienst erweitert. Innerhalb der DMZ und des Workspace-Netzwerkes wurde jeweils ein System hinzugefügt, das den Backup-Dienst bereitstellt. Weitere Details zum Testszenario und insbesondere zu den vorhandenen Flows finden sich in dem Testfall, der die Datenbank für das Testszenario vorbereitet (Klasse: **de.vitsi.module.indiprio.scenario.ScenarioGraphDatabase**, Artefakt ID: Artefakt ID: *de.vitsi.module.indiprio.scenario*).

7.2 Qualität der generierten Prioritäten

Hostname	$value_{conf}$	$value_{integ}$	$value_{avail}$	dam_{tf}	dam_{ltf}
dmzBackup	354	354	70	77	12863
workspaceBackup	354	354	70	77	4847
Server1	2721	10609	9781	15254	903
Server2	1509	3529	3219	825	12281
Server3	4304	2152	1614	807	4870
WS1-WS2					16355
WS3					16194
WS4-WS5					16181
WS6-WS8					15603

Tabelle 7.3: Berechnete Werte der Hosts

Durch die Anwendung der Vorverarbeitung der Eingabeinformationen wurde die in 7.3 dargestellte Tabelle der Eingabewerte für die Priorisierung erzeugt. Interessant ist hierbei vor allem der Unterschied im dam_{ltf} zwischen den beiden Backup-Systemen. Dieser entsteht vor allem dadurch, dass sich das DMZ-Backup-System mit den beiden wichtigen DMZ-Servern in einem Layer-2-Netzwerk befindet.

Beachtenswert ist des Weiteren die starke Beeinflussung des dam_{ltf}-Wertes durch den dam_{tf}-Wert des jeweils anderen Servers. Der dam_{ltf}-Wert des Accounting-Server3 ist deswegen relativ niedrig, weil er im Beispiel-Szenario keine ausgehenden Verbindungen zu anderen Systemen aufgebaut hat.

Die Workstations 1 bis 3, welche sich im Accounting-Netzwerk befinden, haben einen geringfügig höheren dam_{ltf}-Wert als die Workstations 4 bis 8 des Workspace-Netzwerkes. Dass der Unterschied nicht gravierender ausfällt, liegt daran, dass auf dem im Accounting-Netzwerk befindlichen Server keine angreifbare Sicherheitslücke vorhanden ist. Die Variationen der Prioritäten der Workstations innerhalb eines Netzwerkes werden durch unterschiedliche Flow-Informationen begründet. Des Weiteren zeigen die Werte der Workstations, dass durch die Beachtung verschiedener Informationsquellen und deren Kombinationen eine sinnvolle Priorisierung derjenigen Netzwerkkomponenten möglich ist, die selbst keinen besonderen Wert in der Form von bereitgestellten Diensten darstellen.

Um die Verteilung der errechneten Prioritäten im Szenario-Netzwerk beurteilen zu können, wurde ein Histogramm erstellt, das in 7.4 abgebildet ist. Hierbei gibt die x-Scala den Wert der Priorität an, die für einen Test-Alarm errechnet wurde, und die y-Skala die Anzahl, wie oft dieser Prioritätswert einem Alarm zugewiesen wurde. Das Histogramm wurde auf Basis des Szenarios generiert, indem das Kreuzprodukt der verfügbaren Sicherheitslücken mit allen Hosts jeweils als Ziel und Quelle durchgeführt wurde.

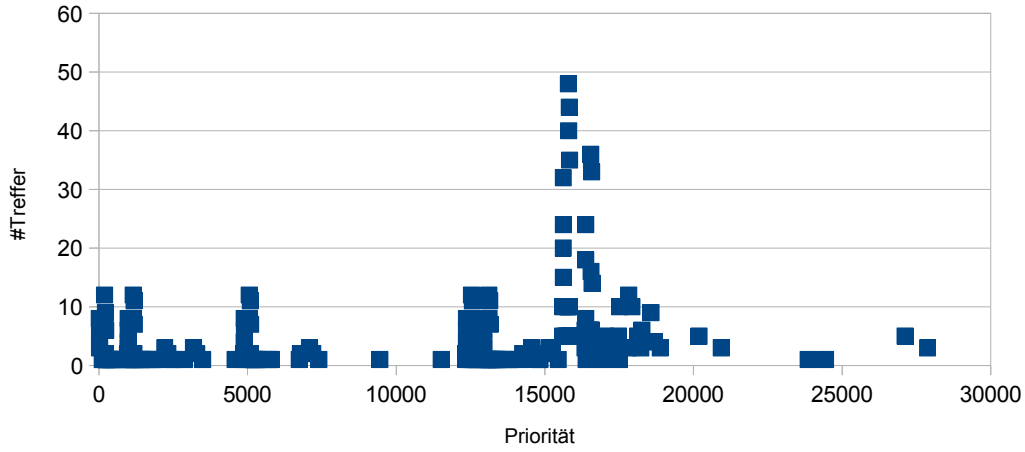


Abbildung 7.4: Szenario Histogramm

Auf diese Art wurden 1456 verschiedene Alarme generiert, von denen 561 auf verschiedene Prioritätswerte abgebildet wurden. Der minimale Prioritätswert lag bei 19, der maximale Wert bei 27827. Der Mittelwert der Prioritäten lag bei 12722,21 bei einer Standardabweichung von 5764,20. Da es in der Diensthierarchie einen Gesamtwert von 30 000 gab, war dies genau im erwarteten Bereich zwischen 0 und 30 000. Es wären zwar auch Werte über 30 000, also der Summe der in der Diensthierarchie vorhandenen Dienstwerte, möglich. Dies könnte aber nur auftreten, wenn die im Unternehmen bereitgestellten Dienste auf wenige Server verteilt würden. Der Grund für diese Grenzen der Prioritätswerte liegt darin, dass über die Gleichungen 4.5 und 4.7 nur die vom Administrator angegebenen $\text{value}_{\text{admin}}$-Werte auf die Dienste in den Kategorien $\text{value}_{\text{conf}}$, $\text{value}_{\text{integ}}$ und $\text{value}_{\text{avail}}$ der untersten Dienstebene verteilt werden. Von dort aus werden sie über 4.9 auf die Dienste bereitstellenden Hosts verteilt. Die so verteilten Werte gehen dann, wie in 7.1 beschrieben, in die Prioritätswerte der Alarme ein.

Da das Beispiel-Szenario allerdings nur Einblicke in das Verhalten der Priorisierung in einem ganz konkreten Fall gibt, sollte die Priorisierungsformel generell auf ihre Eigenschaften untersucht werden.

$$\begin{aligned}
\text{priority}(a) &= \text{dam}_{\text{ltf}}(s) + \text{p}_{\text{vs}}(v) \cdot \text{p}_{\text{ve}}(t,v) \cdot \text{p}_{\text{ha}}(t) \cdot (\, \text{p}_{\text{vc}}(a) \cdot \text{dam}_{\text{ltf}}(t) + \\
&\qquad \text{imp}_{\text{c}}(v) \cdot \text{value}_{\text{conf}}(t) + \text{imp}_{\text{i}}(v) \cdot \text{value}_{\text{integ}}(t) + \text{imp}_{\text{a}}(v) \cdot \text{value}_{\text{avail}}(t)), \\
s &= \text{source}(a), \\
v &= \text{vulnerability}(a), \\
t &= \text{target}(a)
\end{aligned} \tag{7.1}$$

Fügt man die im Kapitel 4.4 vorgestellten Berechnungsvorschriften in eine Formel zusammen und vereinfacht sie, so erhält man den Term 7.1. Hierbei ist noch zu beachten, das alle Eingabewerte, die sich auf Wahrscheinlichkeiten beziehen, also mit p dargestellt sind, als positive Werte zwischen 0 und 1 definiert sind. Dies gilt ebenso für das Maß der Beeinflussung (imp_c, imp_i und imp_a) der vorhandenen Vertraulichkeit, Integrität bzw. Verfügbarkeit. Als Folge der Vorgaben der Graph-Berechnung sind die vorhandenen CIA-Werte ($value_{conf}$, $value_{integ}$ und $value_{avail}$) sowie das Schadenspotential (dam_{ltf}), das ein Host im Falle einer Übernahme im Netzwerk erzeugen kann, ebenso positiv. Die Größe der Werte hängt hierbei natürlich von den Eingaben des Administrators innerhalb der Diensthierarchie und der Anzahl der Dienste bereitstellenden Server ab. Ein weiterer wichtiger Faktor für die Größe der dam_{ltf} ist insbesondere die Struktur der Netzwerke.

Aus der Formel 7.1 lässt sich ersehen, dass diese nur Multiplikationen und Additionen unterschiedlich großer positiver Zahlen enthält. Es kann deshalb nicht zu einer Auslöschung kommen. Somit können sich Rundungsfehler in der Berechnung nicht in einem für die Priorität relevanten Maß aufbauen. Aus der Betrachtung der Ergebnisse der Berechnungsvorschrift kann des Weiteren festgehalten werden, dass die Änderungen der Eingabedaten sich proportional auf die errechnete Priorität auswirken.

Dies gilt auch für alle Gleichungen, die in dieser Arbeit für die Vorverarbeitung der Eingabewerte in Kapitel 4.3 vorgeschlagen wurden.

7.3 Performance

Um die Performance der im Prototypen vorliegenden Implementierung des vorgeschlagenen Priorisierungsalgorithmus zu bewerten, wurde ein Unit-Test geschrieben, der die Prioritätsberechnung mit den gleichen Eingabedaten wiederholt ausführt. Um etwaige Laufzeitoptimierungen zu ermöglichen, wurde erst eine große Anzahl (10 000) von Berechnungen durchgeführt, die nicht in das Ergebnis der Performance-Analyse einflossen, gefolgt von einer großen Anzahl (1 000 000 pro Test) von Berechnungsdurchläufen, deren durchschnittliche Dauer ermittelt wurde. Von der Betrachtung der minimalen und maximalen Werte der Ausführungsdauer wurde abgesehen, da diese zu stark von äußeren Einflüssen verfälscht werden. Die beiden Testsysteme waren mit den folgenden Prozessoren bestückt:

- Intel® Core™ i7-M640@2.80GHz, CPU (2 Cores, 2 Threads/Core)
- AMD FX-8120 (4 Cores, 2 Threads/Core)

Die Tabelle 7.4 stellt die Ergebnisse der Performance-Messung dar (erstellt durch **IndiprioPrioritizerBenchmark**-Klasse). Es konnte somit gezeigt werden, dass die Berechnung eines Alarms auf gängiger Hardware innerhalb von wenigen tausend Na-

Threads	Optimiert (ns)	Skalierung	AMD Standard (ns)	Skalierung	Relativ. Skalierung
1	2886	1,00	6287	1,00	2,18
2	1530	1,89	5320	1,18	3,48
3	1140	2,53	4025	1,56	3,53
4	844	3,42	3526	1,78	4,18
5	795	3,63	3059	2,06	3,85
6	828	3,49	2917	2,16	3,52
7	749	3,85	2652	2,37	3,54
8	765	3,77	2527	2,49	3,30
9	733	3,94	2574	2,44	3,51
10	750	3,85	2606	2,41	3,47
11	764	3,78	2574	2,44	3,37
12	750	3,85	2590	2,43	3,45
13	733	3,94	2574	2,44	3,51
14	749	3,85	2683	2,34	3,58
15	733	3,94	2621	2,40	3,58
16	734	3,93	2559	2,46	3,49

Threads	Optimiert (ns)	Skalierung	Intel Standard (ns)	Skalierung	Relativ. Skalierung
1	2392	1,00	9033	1,00	3,78
2	1253	1,91	5686	1,59	4,54
3	970	2,47	3881	2,33	4,00
4	987	2,42	3672	2,46	3,72
5	922	2,59	4025	2,24	4,37
6	910	2,63	3695	2,44	4,06
7	900	2,66	3963	2,28	4,40
8	917	2,61	3721	2,43	4,06

Tabelle 7.4: Performance-Messungen

nosekunden erfolgen kann, ohne jedoch das Sammeln der notwendigen Daten aus der Datenbank zu beachten. Es wurden zwei Arten der Implementierung der Prioritätsberechnung aus den vorverarbeiteten Eingabedaten vorgenommen. Die „Standard"-Implementierung legt hierbei den Fokus auf ein gutes Software-Design und somit Les- und Wartbarkeit des Sourcecodes. Wohingegen bei der auf Performance optimierten Version die Berechnungsgeschwindigkeit im Fokus lag.

In 7.4 ist pro System, auf dem die Tests ausgeführt wurden, ein Tabellenblatt dargestellt. Die Spalte Threads bezeichnet die für die Berechnung verwendete Anzahl von Threads. Optimiert (ns) bzw Standard (ns) geben die Dauer in Nanosekunden an, die der Test benötigt hat. Die Skalierungsspalten geben zu den jeweils rechts von ihnen angegebenen

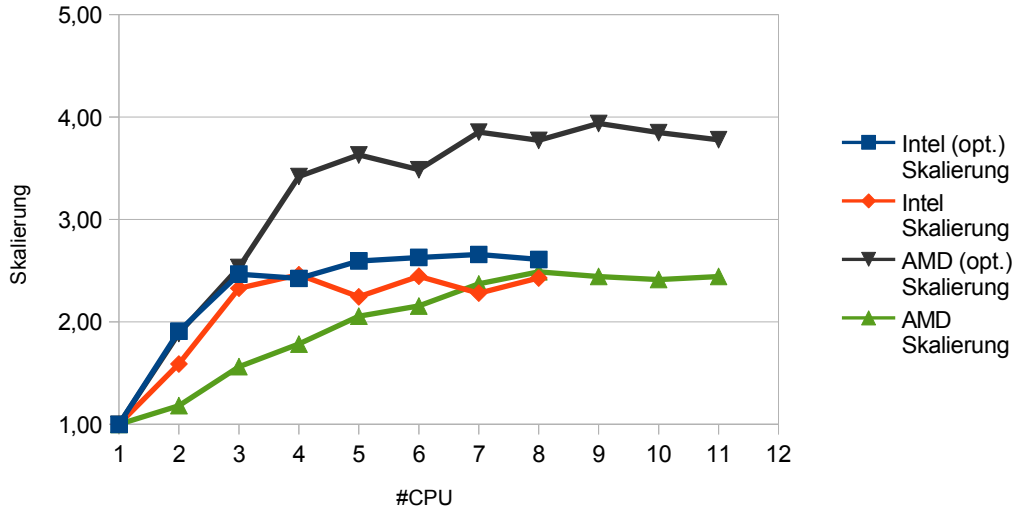


Abbildung 7.5: Skalierung der Prioritätsberechnung

Zeitspalten den Skalierungsfaktor relativ zum Wert mit einem Thread an. Die Relativ. Skalierung gibt die Beschleunigung zwischen der nicht optimierten und der optimierten Implementierung an.

Es wurde außerdem nachgewiesen, dass gegenüber der Standard-Implementierung eine deutliche Steigerung der Performance erreicht werden konnte. Dabei wurde aber ein Verlust der Wartbarkeit des Sourcecodes in Kauf genommen. Aus dem Diagramm 7.5 ist ersichtlich, dass die Skalierung der optimierten Berechnung durch die reduzierten RAM-Zugriffe deutlich besser ausfällt als die nicht optimierte Variante. Außerdem ist ersichtlich, dass die Skalierung beider Berechnungsarten von echten CPU-Kernen wesentlich stärker profitiert als von zusätzlichen Threads pro Kern.

Um zu erkennen, an welcher Stelle weitere Optimierungen sinnvoll sind, wurden außerdem Messungen zur Dauer des Ladens der Indikatoren für den gegebenen Alarm aus der Graphdatenbank durchgeführt. Innerhalb des Szenarios hat sich so gezeigt, dass der Erwartungswert für die Dauer der Ladeoperation von 1465 Messungen bei 2,3 Millisekunden, bei einer Standardabweichung von 1,9 Millisekunden, auf dem im Test verwendeten Intel System lag. Der kleinste gemessene Wert war 1, der größte 64 Millisekunden. (Test-Klasse: **IndiprioSzenarioBenchmark#benchmarkIndicatorProducer**)

Es konnte somit gezeigt werden, dass die Dauer der Berechnung gegenüber der Dauer der Datenbankzugriffe vernachlässigt werden kann. Die Dauer eines Datenbankzugriffs liegt im Bereich von wenigen Millisekunden, wohingegen die Berechnung der Priorität eines Alarmes innerhalb einiger tausend Nanosekunden durchgeführt werden kann.

Zusätzlich zum Test-Szenario wurden weitere Tests durchgeführt, die dazu dienen sollten, zu prüfen, wie sich die *Refinement*-Komponente bei steigender Komplexität des Graphen verhält.

Hierzu wurden abhängig von den Eingabeparametern Netzwerke zufällig generiert. Zwei Durchläufe mit gleichen Eingabeparametern erzeugen somit das gleiche Netzwerk.

Die Diensthierarchie dieses Netzwerkes ist so aufgebaut, dass die spezifizierte Anzahl von Diensten nur die Anzahl der Dienste auf der untersten Ebene angibt. Darauf aufbauend wird eine Diensthierarchie generiert, die pro Ebene so viele Dienste zusammenfasst, dass sich immer drei Ebenen bis zur Wurzel der Diensthierarchie ergeben.

Innerhalb der generierten Systemlandschaft wurden zwei Arten von Netzwerken unterschieden. Die erste Art sind Dienstnetzwerke, in denen Serversysteme stehen. Die zweite Art sind Workspace-Netzwerke, auf die sich die Workspace-Systeme verteilen. Das Routing wurde so aufgebaut, dass jedes Workspace-Netzwerk einen Zugriff auf zwei Dienstnetzwerke erhalten hat. Jedes Dienstnetzwerk hat des Weiteren einen Layer-3-Zugriff auf einige andere Dienstnetzwerke erhalten.

In der Grundkonfiguration enthält die generierte IT-Landschaft 100 Dienste auf der untersten Dienstebene mit einer dreistufigen Hierarchie bis zum Wurzelknoten. Diese Dienste werden auf fünf Server in fünf Dienstnetzwerken abgebildet. Des Weiteren gibt es 400 Workstations, die sich auf 20 Workstation-Netzwerke verteilen (somit 20 Workstations pro Netzwerk).

Bei den Berechnungen wurden jedes Mal 25 Berechnungen gemessen, deren zeitliche Ergebnisse verworfen wurden und danach 100 Messungen, die in die Statistiken eingerechnet wurden.

Die Messung der Dauer der Vorverarbeitungsberechnung wurde hierbei in drei aufeinanderfolgende Schritte unterteilt:

TN.1 Berechnung der Werte der Dienste über die Diensthierarchie.

TN.2 Berechnung der Host-dam_{tf}-Werte.

TN.3 Berechnung der Host-dam_{ltf}-Werte.

In jedem der folgenden Tests wurde jeweils ein Aspekt des Testnetzwerks variiert, um zu messen, wie sich diese Änderung alleine auf die Performance der Vorverarbeitung der Eingabewerte auswirkt. Die Tests sind:

RT.1 Test der Performance des Refinements bei einer steigenden Anzahl an Diensten und einer konstanten Anzahl an Dienste bereitstellenden Servern und Tiefe der Diensthierarchie.

RT.2 Test des Performance Verhaltens bei einer steigenden Anzahl von Workstations und einer konstanten Anzahl von Workstations pro Workstation-Netzwerk.

RT.3 Test der Performance des Refinements bei einer steigenden Anzahl von Workstations pro Workstation-Netzwerk, wobei die Gesamtzahl der Workstations konstant war.

In den folgenden Beschreibungen der Testergebnisse wird nur auf die Messwerte aus RT.1 - RT.3 in Bezug auf TN.1 - TN.3 eingegangen, die eine Abhängigkeit zu den Eingabedaten gezeigt haben.

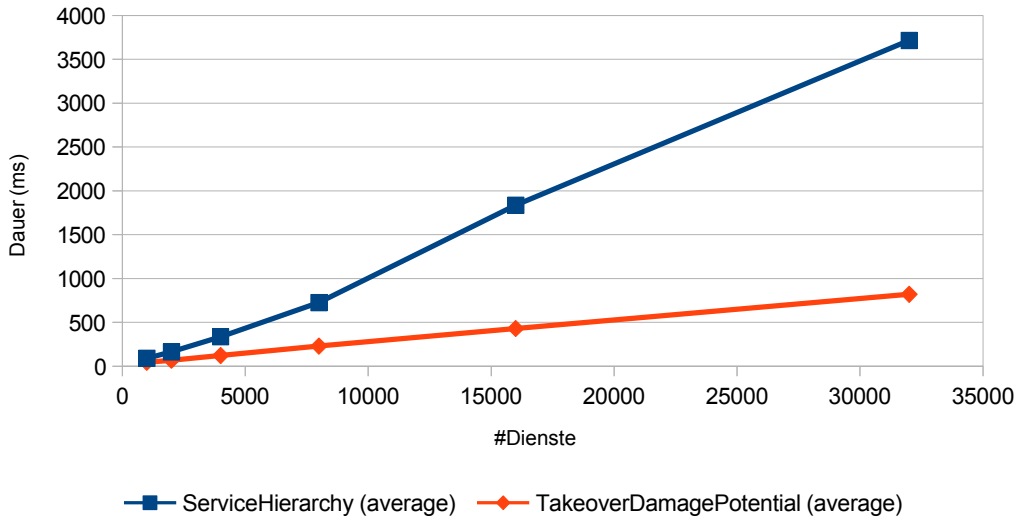


Abbildung 7.6: Dauer der Berechnungen des Dienstwertes.

Im Test RT.1 wurde die Steigerung der Komplexität der Diensthierarchie exemplarisch gemessen. Im Diagramm 7.6 ist dargestellt, wie sich die Dauer der Berechnung von TN.1 und TN.2 in Abhängigkeit von der Anzahl der Dienste verhält. Innerhalb der ausgeführten Tests wurde somit eine lineare Steigung in Abhängigkeit von der Anzahl der bereitgestellenten Dienste gemessen. Hierbei wurde die Anzahl der Dienste bereitstellenden Server bewusst konstant gehalten. Es sollte dadurch der Effekt, der innerhalb der Berechnungen durch eine steigende Anzahl von Servern auftritt, eliminiert werden. Der Grund hierfür ist, dass dies der gleiche Effekt ist, der bei der Steigerung der Anzahl von Workstations auftritt und somit bereits im Test RT.2 abgebildet ist.

Im Test RT.2, bei dem die Anzahl der Workstations und mit ihr die Anzahl der Workstation-Netzwerke erhöht wurde, sind insbesondere die Ergebnisse von TN.2 und TN.3 interessant. Die Anzahl der Workstation-Netzwerke musste erhöht werden, um eine konstante Anzahl von Workstations pro Netzwerk zu erreichen. Die Ergebnisse sind in den Abbildungen 7.7 und 7.8 dargestellt. Auch hier lässt sich eine angenähert lineare Abhängigkeit zur Erhöhung der Anzahl der Workstations im Testnetzwerk ermitteln.

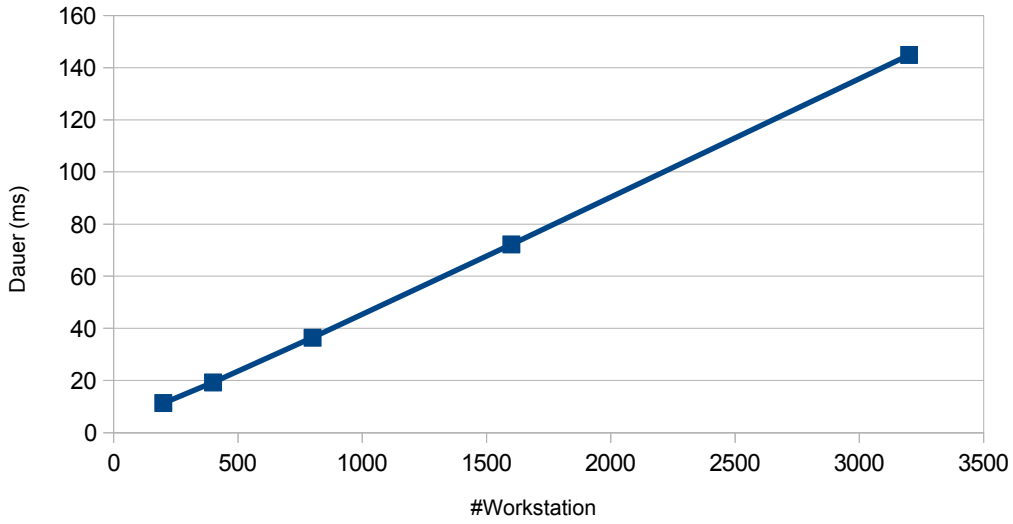


Abbildung 7.7: Performance der Berechnung dam_{tf}.

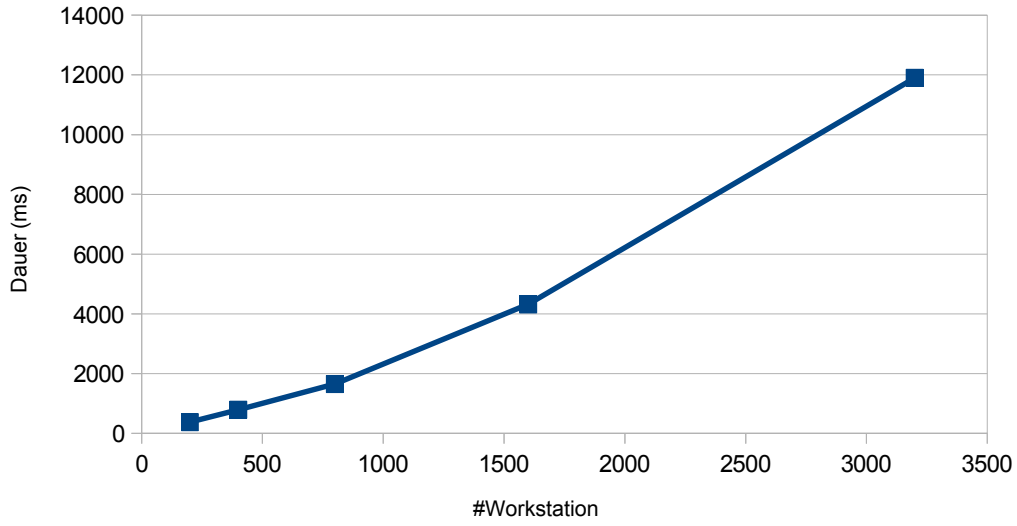


Abbildung 7.8: Performance der Berechnung dam_{ltf} abhängig von der Anzahl der Workstations.

Im letzten Refinement-Test RT.3, bei dem ausschließlich die Anzahl der Workstation-Netzwerke variiert wurde, konnte ebenso, wie in 7.9 dargestellt, eine lineare Abhängigkeit der Dauer der Berechnung von TN.3 von der Größe der Netzwerke festgestellt werden.

Bei der Betrachtung der Performance-Werte ist jedoch zu beachten, dass die hier durchgeführten Berechnungen sich auf eine vollständige Neuberechnung aller Werte der

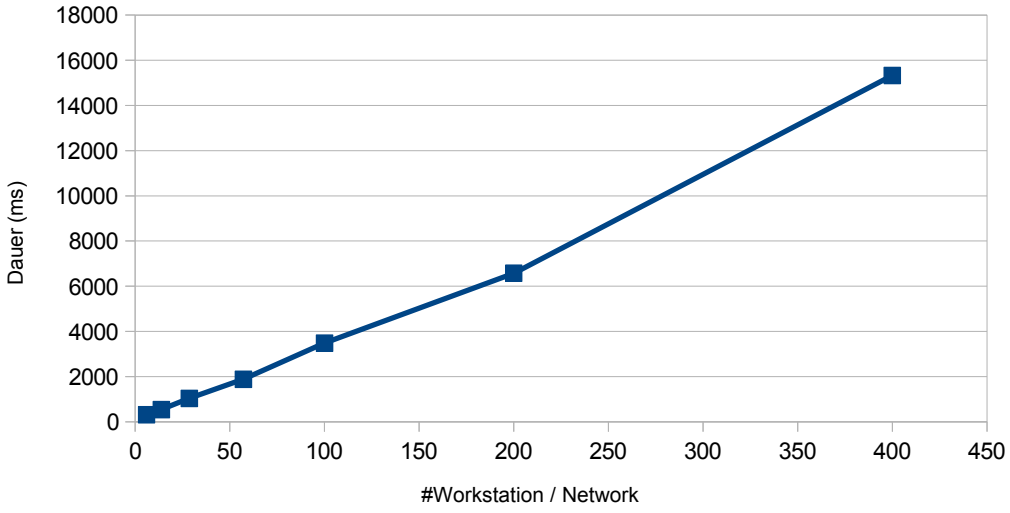


Abbildung 7.9: Performance der Berechnung dam_{ltf} abhängig von der Netzwerkgröße

Dienste bzw. Hosts beziehen. Dies ist jedoch nur in wenigen Fällen notwendig, da meist nur wenige Hosts von einer Änderung betroffen sind. Beispielsweise müssen beim Bekanntwerden eines neuen Flows nur die vorverarbeiteten Werte der beiden Hosts angepasst werden und nicht die des gesamten Netzwerks.

Die zwei Fälle, in denen alle vorverarbeiteten Werte angepasst werden müssen, sind das Hinzukommen bzw. Entfernen eines Dienstes und die Änderung der Werte $\text{confidentiality}_{\text{admin}}$, $\text{integrity}_{\text{admin}}$ und $\text{availability}_{\text{admin}}$ eines Dienstes. In allen anderen Fällen muss nur ein Bruchteil der Werte angepasst werden.

Das Priorisierungssystem ist deshalb auch für den Einsatz in großen Netzwerken geeignet, insbesondere weil sich die benötigte Rechenleistung linear zu der Steigerung der Komplexität der Eingabedaten verhält.

7.4 Vergleich der Ansätze

Nachdem der entwickelte Priorisierungsansatz evaluiert wurde, kann er nun gemäß den Anforderungen 2.8 und den Kriterien aus 2.7 bewertet werden.

Die Bewertung der Qualität der Priorisierung in 7.2 zeigt, dass die Ergebnisse des vorliegenden Priorisierungsansatzes der tatsächlichen Kritikalität des Alarmes für das zu schützende Netzwerk entsprechen und somit die Anforderung A.1 durch den Ansatz erfüllt wird. Die Zusammenführung der Informationsquellen in der graphbasierten Struktur ermöglicht es, die Informationen sowohl in einem für den Menschen als auch

für die Software interpretierbaren Format bereitzustellen und somit zu der Transparenz der Prioritätsberechnung beizutragen.

Da der Priorisierungsansatz auf grundlegenden Datenquellen basiert, die in jedem Netzwerk erfasst werden können, erfüllt er die Anforderung A.2. Der Ansatz ist somit in allen Netzwerken einsetzbar und flexibel.

Wie die Evaluierung des Prototyps gezeigt hat, ist die Aufteilung der Prioritätsberechnung in zwei Phasen für die Skalierbarkeit des Systems von großem Nutzen. Somit wird die Anforderung A.3 zur Skalierbarkeit durch den Priorisierungsansatz erfüllt. Dadurch wird sichergestellt, dass das Priorisierungsystem auch in Hochlast-Situationen eines Angriffs gut funktionieren kann. Es ist somit unwahrscheinlich, dass die Berechnung der Priorisierung zu einer Verzögerung der Verarbeitung der Alarme führt.

Die Anforderung A.4 für die Automatisierung des Priorisierungssystems kann somit als erfüllt angesehen werden.

Wie in Kapitel 3 beschrieben, ist eine automatisierte Erfassung der Informationen aus den für diesen Ansatz ausgewählten Quellen im regulären Betrieb möglich. Dadurch kann auch der Betrieb des Priorisierungssystems automatisiert werden.

Zur Inbetriebnahme (Setup) des Systems ist zu sagen, dass nicht alle manuellen Eingaben der Administratoren eliminiert werden konnten. Die vollständige Ersetzung des Expertenwissens durch messbare Werte gestaltet sich schwierig. Durch die Verbindung der manuell spezifizierten Werte mit messbaren Werten konnten diese Eingaben jedoch auf einen Bereich verlagert werden, der für Administratoren verständlich ist und sich nur sehr selten ändert.

Dank der Verwendung eines einfachen und somit verständlichen Berechnungsschemas für die Prioritäten ist es möglich, dieses Priorisierungssystem einzusetzen, ohne Fachkenntnisse erwerben zu müssen, die über grundlegende mathematische Kenntnisse hinausgehen.

Da sich die Granularität und der Wertebereich des vorgeschlagenen Systems an den Werten im Unternehmen orientiert, wird die Granularität der erzeugten Prioritäten automatisch an das gegebene System angepasst. Somit ist die Anforderung A.5 erfüllt.

Da die Verteilung der Eingabeparameter eine wichtige Rolle für die Verteilung der Priorisierung in dem entwickelten Berechnungsschema spielt, wäre eine bessere Verteilung in den Eingabewerten wünschenswert. Insbesondere wäre eine feingranularere Spezifikation der Impact-Werte des CVSS von Vorteil.

Um möglichst gute Priorisierungswerte zu erhalten, sollten entweder alle Hosts des zu schützenden Systems eigene Dienste erbringen oder der in 4.3 vorgeschlagene alternative Ansatz zur Erfassung der CIA-Host-Werte für die keine Dienste erbringenden Hosts umgesetzt werden.

Ein zusätzlicher Vorteil des Ansatzes ist, dass die berechneten Eingabewerte der Priorisierung einen guten Einblick in die Sicherheit eines Systems geben. Insbesondere der dam_{ltf}-Wert der einzelnen Hosts gibt einen Einblick, wie groß die Gefährdung für das gesamte Netzwerk durch einen einzelnen Host ist und zeigt Stellen des Netzwerkes auf, an denen besondere Sicherungsmaßnahmen angebracht sind.

	RIM	FuzMet	Attack Graph	Bayes Networks	Hidden Markov	CVSS (v2)	IndiPrio
Flexibilität	+	+	-	+	+	-	+
Skalierbarkeit	+	+	-	+	-	+	+
Automatisierbarkeit	-	-	+	o	+	-	+
Setup	-	-	-	-	-	o	o/+
Granularität	+	+	o	+	+	o	+

Tabelle 7.5: Vergleich der Ansätze

Diese Bewertung der Eigenschaften des in dieser Arbeit entwickelten IndiPrio-Ansatzes kann dazu verwendet werden, die bereits aus 2.7 bekannte Vergleichstabelle, wie in 7.5 dargestellt, zu erweitern.

7.5 Zusammenfassung

Zusammenfassend lässt sich sagen, dass die Evaluierung ein positives Ergebnis geliefert hat. Es konnte insbesondere die Erfüllung der gestellten Anforderungen gezeigt werden. Der entwickelte Ansatz zeigt mehrere Vorteile auf, die zu einer positiven Bewertung der wichtigsten Eigenschaften in der Vergleichstabelle führen.

Kapitel 8

Zusammenfassung & Ausblick

Am Anfang dieser Arbeit wurde die Problematik beschrieben, dass IDS-Systeme oft innerhalb kürzester Zeit eine Vielzahl von Alarmen generieren. Die dadurch möglicherweise entstehende Verwirrung könnte verzögerte oder gar falsche Reaktionen bewirken. Als Möglichkeit, dies zu verhindern und den durch die Angriffe erzeugten Schaden zu minimieren, könnte die Priorisierung der Alarme dienen, sofern sie eine zeitnahe Abarbeitung der wichtigsten Ereignisse sicherstellt. Ob und wie dies gewährleistet werden kann, war das Ziel des dieser Arbeit zugrunde liegenden Forschungsprojekts. Zur Strukturierung der Arbeit an dessen Problemstellung wurden drei wissenschaftliche Fragen formuliert und beantwortet.

Damit der Einstieg in die Entwicklung eines Erfolg versprechenden Priorisierungssystems gefunden werden konnte, wurden zunächst bereits bestehende Ansätze auf ihre Eigenschaften hin analysiert und Anforderungen identifiziert, die der zu entwickelnde Priorisierungsansatz erfüllen sollte.

Als Nächstes wurden einige Informationsquellen dargestellt und sowohl die von ihnen bereitgestellten Informationen als auch ihre Einsatzmöglichkeiten bei der Priorisierung von Alarmen beleuchtet. Eine wichtige Rolle spielte hierbei die Darstellung der Wege zur automatisierten Informationserfassung. Nach dieser Beschreibung wurden einige Methoden und Techniken aufgezeigt, wie die gewonnenen Daten der Informationsquellen kombiniert werden können.

Aufbauend auf dem gewonnenen Wissen über die Struktur der relevanten Informationen wurde ein Graph-basierter Ansatz für die Zusammenführung der Informationen entwickelt. Mit diesem Graphen als Grundlage wurde die Berechnungsvorschrift zur Ermittlung der Priorität eines Alarmes ausgearbeitet. Diese wurde in zwei Schritte, die Vorverarbeitung der Eingabewerte und die auf diesen Werten aufbauende Prioritätsberechnung, aufgeteilt.

Basierend auf dieser Aufteilung und den verwendeten Informationsquellen wurde eine

Modulstruktur und das für die Verarbeitung der Informationen notwendige Datenmodell definiert. Zusätzlich wurden die Prozesse beschrieben, die für die Berechnung der Prioritäten ausgeführt werden müssen.

Um die Umsetzbarkeit der Priorisierung nachzuweisen und deren Eigenschaften evaluieren zu können, wurden die wichtigsten Module in Form eines Prototypen innerhalb des VITSI-Systems umgesetzt. Deren Funktionalität und die Auswahl der für den Prototypen verwendeten Techniken wurde ebenso beschrieben.

Der so entstandene Prototyp und die zugrundeliegenden Formeln wurden als Nächstes einer Evaluierung auf ihre Eigenschaften hin unterzogen. Dabei lagen die spezifizierten Anforderungen und der Vergleich zu den bestehenden Ansätzen im Fokus. Als Ergebnis konnte festgehalten werden, dass die gestellten Anforderungen durch den entwickelten Ansatz erfüllt werden und dessen Bewertung gemäß den definierten Kriterien positiv ausfällt.

Um die Bewertung des Ansatzes zu vertiefen, ist es sinnvoll, den entwickelten Prototypen in das am Lehrstuhl für Netzarchitekturen und Netzdienste entwickelte Testbett zu integrieren.

Sollte eine weitere Performance-Optimierung erforderlich sein, könnte diese erreicht werden, indem die für die Priorisierung verwendeten vorverarbeiteten Informationen innerhalb eines Caches in den Hauptspeicher des priorisierenden Systems geladen werden. Auf diese Weise ließen sich die teuren Datenbankoperationen während der Priorisierung vermeiden. Damit wäre eine weitere deutliche Performance-Steigerung möglich.

Innerhalb der Berechnung der Priorität des Alarms besteht weiterer Spielraum für eine Optimierung. Es könnte auf die Speicherung der Zwischenwerte sowie der maximal und minimal möglichen Priorität verzichtet werden. Auf diese Weise wäre es möglich, die Dauer der Prioritätsberechnung auf die Größenordnung einiger hundert *Java*-Methodenaufrufe zu reduzieren. Sollten diese Werte später für die Inspektion des Alarms durch einen Administrator benötigt werden, könnten sie dann aus den Eingabedaten errechnet werden.

Die Netzwerke und IDS-Systeme werden permanent weiterentwickelt. Das Gleiche gilt natürlich auch für die Methoden und Tools der Angreifer. Folglich wären in Zukunft weitere Parameter in die Berechnung der Prioritäten einzubeziehen, um die Verteilung der Prioritätswerte zu verbessern.

Anhang A

Anhang

A.1 Performance-Werte des Refinements

Services	Servers	Service Networks	Workstations	Workstation Networks	Workstations / Network	warmup (c)	counting (c)	ServiceHierarchy (average,ms)	ServiceHierarchy (stddev,ms)	Relative	TakeoverDamageP otential (average,ms)	TakeoverDamageP otential (stddev,ms)	Relative	LocalizedTakoverD amageP otential (average,ms)	LocalizedTakoverD amageP otential (stddev,ms)	Relative
1000	5	400	20	20	25	100	89,54	22,66	25,31%	44,64	8,29	18,57%	751,68	39,39	5,24%
2000	5	400	20	20	25	100	164,76	15,4	9,35%	68,75	2,81	4,09%	808,74	69,31	8,57%
4000	5	400	20	20	25	100	335,67	37,13	11,06%	122,15	6,52	5,34%	739,16	9,51	1,29%
8000	5	400	20	20	25	100	725,88	75,65	10,42%	230,31	4,98	2,16%	788,1	10,88	1,38%
16000	5	400	20	20	25	100	1836,07	461,74	25,15%	430,47	5,32	1,24%	786,66	9,07	1,15%
32000	5	400	20	20	25	100	3714,75	413,91	11,14%	820,27	23,13	2,82%	774,97	23,52	3,03%
100	5	200	10	20	25	100	8,78	1,91	21,75%	11,36	1,34	11,80%	373,83	9,82	2,63%
100	5	400	20	20	25	100	8,6	1,12	13,02%	19,3	1,69	8,76%	789,25	12,74	1,61%
100	5	800	40	20	25	100	9,06	3,67	40,51%	36,45	5,12	14,05%	1651,52	17,64	1,07%
100	5	1600	80	20	25	100	9,16	4,35	47,49%	72,2	9,5	13,16%	4318,3	75,31	1,74%
100	5	3200	160	20	25	100	8,44	1,08	12,80%	144,82	6,98	4,82%	11898,11	125,89	1,06%
100	5	400	67	5,97	25	100	8,94	4,6	51,45%	19,12	1,52	7,95%	309,77	16,49	5,32%
100	5	400	29	13,79	25	100	8,57	1,28	14,94%	19,11	1,57	8,22%	537,44	5,76	1,07%
100	5	400	14	28,57	25	100	8,44	1,06	12,56%	19,27	3,9	20,24%	1035,15	31,96	3,09%
100	5	400	7	57,14	25	100	8,95	3,1	34,64%	18,99	1,79	9,43%	1878,39	30,9	1,65%
100	5	400	4	100	25	100	8,47	1,12	13,22%	19,15	1,72	8,98%	3479,61	66,83	1,92%
100	5	400	2	200	25	100	8,44	0,96	11,37%	19,46	3,34	17,16%	6570,75	116,62	1,77%
100	5	400	1	400	25	100	8,66	1,48	17,09%	19,37	1,63	8,42%	15325,32	306,47	2,00%

Tabelle A.1: Performance-Werte des Refinements auf dem Intel-System.

A.2 Abkürzungsverzeichnis

Name	Abkürzung
Affero General Public License	AGPL
Analytischer Hierarchieprozess	AHP
Business Process Model and Notation	BPMN
Context Dependency Injection	CDI
Collateral Damage Potential	CDP
Confidentiality, Integrity and Availability	CIA
Common Vulnerabilities and Exposures	CVE
Common Vulnerability Scoring System	CVSS
Demilitarized Zone	DMZ
Domain Name System	DNS
High Speed Network Monitoring and Analysis	HISTORY
Hidden-Markov-Modells	HMM
Infrastructure as a Service	IaaS
Intrusion Detection Message Exchange Format	IDMEF
Intrusion Detection Systemen	IDS
Internet Engineering Task Force	IETF
Internet Information Server	IIS
Internet Protocol Flow Information Export	IPFIX
International Organization for Standardization	ISO
Java Architecture for XML Binding	JAXB
Java Management Extensions	JMX
Java Virtual Machine	JVM
Network Address Translation	NAT
Network Intrusion Detection Systeme	NIDS
National Institute of Standards and Technology	NIST
Network Mapper	NMAP
National Vulnerability Database	NVD
Packet Sampling	PSAMP
Risk Index Model	RIM
Topological Vulnerability Analysis	TVA
Unified Modeling Language	UML
Versatile Monitoring Toolkit	Vermont
Verteilte IT Sicherheitsinfrastruktur	VITSI

Tabelle A.2: Abkürzungsverzeichnis

Literaturverzeichnis

[1] N. B. Anuar, M. Papadaki, S. Furnell, and N. Clarke, "Incident prioritisation using analytic hierarchy process (ahp): Risk index model (rim)," Security and communication networks, vol. 6, no. 9, pp. 1087–1116, 2013.

[2] P. D. habil. A. Grauel. (2014, December) Computational intelligence fuzzy-tutorial. [Online]. Available: http://www4.fh-swf.de/media/downloads/fbin/download_4/lehmann_1/internci2/ci2/skripte/fuzzy_3/Fuzzy-Tutorial_1_Gra_Leh30Teil.pdf

[3] M. Wagner, "Bayes-netze eine einführung," 2000.

[4] P. A. Porras, M. W. Fong, and A. Valdes, "A mission-impact-based approach to infosec alarm correlation," in Recent Advances in Intrusion Detection. Springer, 2002, pp. 95–114.

[5] A. Årnes, F. Valeur, G. Vigna, and R. A. Kemmerer, "Using hidden markov models to evaluate the risks of intrusions," in Recent Advances in Intrusion Detection. Springer, 2006, pp. 145–164.

[6] S. Noel and S. Jajodia, "Optimal ids sensor placement and alert prioritization using attack graphs," Journal of Network and Systems Management, 2008.

[7] H. Debar, D. A. Curry, and B. S. Feinstein, "The intrusion detection message exchange format (idmef)," 2007.

[8] (2014, Mai) 145 millionen kunden von ebay-hack betroffen. [Online]. Available: http://www.heise.de/security/meldung/145-Millionen-Kunden-von-eBay-Hack-betroffen-2195974.html

[9] T. L. Saaty, "Decision making with the analytic hierarchy process," International journal of services sciences, vol. 1, no. 1, pp. 83–98, 2008.

[10] N. B. Anuar, S. Furnell, M. Papadaki, and N. Clarke, "A risk index model for security incident prioritisation," 2011.

[11] K. Alsubhi, I. Aib, and R. Boutaba, "Fuzmet: a fuzzy-logic based alert prioritization engine for intrusion detection systems," International Journal of Network Management, vol. 22, no. 4, pp. 263–284, 2012.

[12] L. A. Zadeh, “Fuzzy sets,” Information and control, vol. 8, no. 3, pp. 338–353, 1965.

[13] M. W. Fong, P. A. Porras, and A. D. J. Valdes, “Prioritizing bayes network alerts,” May 27 2008, uS Patent 7,379,993.

[14] P. Mell, K. Scarfone, and S. Romanosky, “A complete guide to the common vulnerability scoring system version 2.0,” in Published by FIRST-Forum of Incident Response and Security Teams, 2007, pp. 1–23.

[15] A. O. Ibidapo, P. Zavarsky, D. Lindskog, and R. Ruhl, “An analysis of cvss v2 environmental scoring,” in Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom). IEEE, 2011, pp. 1125–1130.

[16] R. B. S. Carsten Eiram and O. S. F. Brian Martin, “Cvss-shortcomings faults and failures,” Jan. 2014. [Online]. Available: https://www.riskbasedsecurity.com/reports/CVSS-ShortcomingsFaultsandFailures.pdf

[17] A. A. Markov, “An example of statistical investigation of the text eugene onegin concerning the connection of samples in chains,” Science in Context, vol. 19, no. 04, pp. 591–600, 2006.

[18] B. Poppinga, “Beschleunigungsbasierte 3d-gestenerkennung mit dem wii-controller,” Universität Oldenburg, 2007.

[19] T. Birmili, P. Wahju, and D. Willig, “Vergleich von iaas-anbietern,” 2012.

[20] ISO/IEC. (2015, Feb.) Information technology - open systems interconnection - basic referencemodel: The basic model. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip

[21] (2015, Feb.) Nmap security scanner. [Online]. Available: https://nmap.org/

[22] (2015, Feb.) Opennms. [Online]. Available: http://www.opennms.org/

[23] (2015, Feb.) Linkd. [Online]. Available: http://www.opennms.org/wiki/Linkd#Discovery_Using_Bridge_Forwarding_Table.2C_Spanning_Tree_Information_and_Ip_Net_To_Media_Table

[24] (2015, Feb.) Openstack. [Online]. Available: https://www.openstack.org/

[25] “Chef,” Feb. 2015. [Online]. Available: https://www.chef.io/

[26] “Puppet,” Feb. 2015. [Online]. Available: http://puppetlabs.com/

[27] (2015, Feb.) Understanding windows automatic updating. [Online]. Available: http://windows.microsoft.com/en-us/windows/understanding-windows-automatic-updating#1TC=windows-7

[28] “Openvas,” Feb. 2015. [Online]. Available: http://www.openvas.org/

[29] “Nessus,” Feb. 2015. [Online]. Available: http://www.tenable.com/products/nessus

[30] R. Steuerwald, “Anbindung von open vulnerability assessment system (openvas) an eine metadata access point (map)-infrastruktur,” 2011.

[31] F. Dressler, R. German, and P. Holleczek, “Selbstorganisierende netzwerksensoren und automatisierte ereigniskorrelation,” in BSI-Workshop IT-Frühwarnsysteme, Bonn, Germany, 2006, pp. 117–128.

[32] B. Claise, “Packet sampling (psamp) protocol specifications,” 2009.

[33] E. B. Claise, “Rfc5101: Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information,” 2008.

[34] “Cve,” Feb. 2015. [Online]. Available: https://cve.mitre.org/

[35] (2015, Feb.) Nvd feed. [Online]. Available: http://nvd.nist.gov/schema/nvd-cve-feed_2.0.xsd

[36] (2015, Feb.) Nist nvd. [Online]. Available: http://nvd.nist.gov/

[37] (2015, Feb.) Ansii-projekt. [Online]. Available: http://www.ansii-projekt.de/

[38] (2015, Feb.) Java 8 openjdk. [Online]. Available: http://openjdk.java.net/projects/jdk8/features

[39] (2015, Feb.) Neo4j. [Online]. Available: http://neo4j.com/

[40] (2015, Feb.) Neo4j-lizenz. [Online]. Available: https://github.com/neo4j/neo4j

[41] (2015, März) Graph aware ltd. [Online]. Available: http://graphaware.com/

[42] (2015, Feb.) Spring data neo4j. [Online]. Available: http://projects.spring.io/spring-data-neo4j/

[43] (2015, Feb.) Jaxb. [Online]. Available: https://jaxb.java.net/

[44] (2015, Feb.) Cdi. [Online]. Available: http://docs.oracle.com/javaee/6/tutorial/doc/giwhl.html

[45] (2015, Feb.) Weld. [Online]. Available: http://weld.cdi-spec.org/

[46] (2015, Feb.) Junit. [Online]. Available: http://junit.org/

[47] (2015, Feb.) Maven. [Online]. Available: http://maven.apache.org/