universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Cybersecurity Decision Support - Ein betriebswirtschaftliches Cyber Security Entscheidungsmodell im Umfeld des Microsoft Betriebssystems“

verfasst von / submitted by

Stefan Kolbinger BA, MA

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 926
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Dr. Gerald Quirchmayr

Abstract

Cyberangriffe nehmen zu und bedrohen die kritische Infrastruktur sowie ganze Wirtschaftszweige. APT Angriffe stellen hierbei besonders gefährliche und mächtige Cyber Angriffe dar. Um sich schützen zu können, wird die informationsunterstützende Analyse und Klassifizierung von Angriffen immer wichtiger. Dies betrifft sowohl die Vorbereitungsphase auf einen Angriff als auch das Incident Handling. Allerdings ist ein bedeutender Anteil an Organisationen auf Cyberangriffe unzureichend vorbereitet. IT Security Mitarbeiter sind mit einer Informationsflut konfrontiert und darauf basierend Entscheidungen zu treffen wird immer komplexer. Um möglichst sichere Entscheidungen treffen zu können, haben sich in der BWL Entscheidungsmodelle bewährt sowie der OODA Loop in der IT Security Domäne etabliert. Das Anliegen der vorliegenden Masterarbeit ist es, Entscheidungsmodelle zu kombinieren und mit dem OODA Loop zu vereinen, um daraus ein für die IT Security Domäne relevantes Entscheidungsmodell abzuleiten. Dieses Entscheidungsmodell wurde im Anschluss als Prototyp implementiert, um bestimmte APT Angriffe anhand der MITRE ATT&CK Wissensdatenbank zu identifizieren und bewährte Gegenmaßnahmen vorzuschlagen. Dadurch kann der Prototyp nicht nur in der Detect Phase des NIST Frameworks von Vorteil sein, sondern auch in der Respond Phase, wenn Entscheidungen benötigt werden, um Gegenmaßnahmen einzuleiten. Nachdem der Prototyp die Observe/Orient Phase des OODA Loops gut unterstützt, kann er ein Security Operations Center systematisch unterstützen. Das vorliegende Entscheidungsmodell kann aufgrund seiner kompakten und doch präzisen Gestaltung wertvolle erste Anhaltspunkte und eine Orientierung für das Handling von APT Cyberattacken liefern und ist daher ein Beitrag zur Unterstützung des Incident Handling.

Keywords: Cybersecurity, Decision Support System, Entscheidungstheorie.

Inhaltsverzeichnis

Abstract i

Abbildungsverzeichnis v

Tabellenverzeichnis vii

1. Einleitung 1
- 1.1. Motivation und Ausgangssituation . . . 1
- 1.2. Problemstellung . . . 2
- 1.3. Struktur der Masterarbeit . . . 5

2. Entscheidungstheorie und Entscheidungsmodelle 7
- 2.1. Grundlagen der Entscheidungstheorie . . . 7
- 2.2. Decision Support Systeme . . . 8
- 2.3. Organisatorische Entscheidungsfindung . . . 9
- 2.4. Entscheidungsmodelle . . . 11

3. Integriertes Entscheidungsmodell 22
- 3.1. Verwendung der vorgestellten Entscheidungsmodelle . . . 22
- 3.2. Beschreibung des Modells . . . 23

4. Use Cases und Requirements des Prototyps 28
- 4.1. Use Cases . . . 28
- 4.2. Requirements . . . 36
  - 4.2.1. Funktionale Requirements . . . 36
  - 4.2.2. Nichtfunktionale Requirements . . . 37

5. Design des Client-Prototyps 38
- 5.1. Geplante Nutzung des Clients . . . 38
- 5.2. Grundstruktur des Clients . . . 38
  - 5.2.1. Beschreibung der System-Komponenten . . . 40
  - 5.2.2. Domain Model des Clients . . . 42
  - 5.2.3. Controller-Klassen . . . 45
  - 5.2.4. View-Methoden . . . 47
- 5.3. Hochladen und Auswerten von Logdateien . . . 47
  - 5.3.1. Logformate als Input . . . 48
  - 5.3.2. YARA-Regeln zur Logauswertung . . . 48
  - 5.3.3. Auswertungsergebnisse im Client . . . 50

5.4. Darstellung der Analyseergebnisse und Feedback 51
5.4.1. Darstellung im Client 52
5.4.2. Darstellung in einer Falldokumentationsdatei 52
5.4.3. Feedback zu Gegenmaßnahmen 53

6. Implementierung **54**
6.1. Verwendeter Technologiestack 54
6.1.1. Programmiersprachen und Entwicklungsumgebung . . . 54
6.1.2. Libraries und Frameworks 54
6.2. Architektur und Deployment 58
6.3. Ausführung des Clients 60
6.4. Aufbau und Struktur des Quellcodes 60
6.5. Benutzeroberfläche 64
6.6. Sicherheit der Anwendung 66

7. Evaluierung **68**
7.1. Tests des Prototyps 68
7.1.1. Test 1: Turla 69
7.1.2. Test 2: APT19 70
7.1.3. Test 3: APT28 72
7.2. Generelle Resultate der Evaluierung 73
7.2.1. Granularität der Analyse 73
7.2.2. Verfügbarkeit von Daten 74
7.2.3. Identifikation von Angriffen 75
7.3. Empfehlungen aus der Evaluierung 75

8. Conclusio **76**
8.1. Modell und Prototyp als Ausgangspunkte für Weiterentwicklung 76
8.1.1. Interface 76
8.1.2. Inputquellen 77
8.1.3. Auswertungsalgorithmus 77
8.1.4. Fallevaluierung durch User 78
8.1.5. Deployment 79
8.2. Zentrale Schlussfolgerungen 79

A. Ergänzte Einträge in Logs generierter PCAPs für Tests 1–3 **82**
A.1. Test 1: Turla 82
A.2. Test 2: APT19 82
A.3. Test 3: APT28 83

Literaturverzeichnis **85**

Abbildungsverzeichnis

1.1. Marktanteil von Betriebssystemen bei Desktop-PCs 2013–2023, nach Monat [sta]. 5

2.1. Logische Prozesse mit sieben Schritten des RDM-Modells (Rational Decision Making) [LLPE10, S.77]. 13
2.2. Entscheidungsprozess des rationalen Modells [Riz14, S.214]. . . . 13
2.3. Entscheidungsprozess des Bounded Rationality Models [Riz14, S.215]. 15
2.4. Organisatorische Entscheidungsfindung als iterative Sequenz, motiviert durch Diagnose und unterbrochen durch Ereignisse [MRT76, S.273]. 17
2.5. OODA-Loop [Boy18, S.384]. 18
2.6. Recognition Primed Decision Model [Kle98, S.25]. 20
2.7. Integrierte Version des Recognition Primed Decision Models [Kle98, S.26]. 21

3.1. Integriertes Entscheidungsmodell 24

4.1. Use-Case-Diagramm des Client 29

5.1. Komponentendiagramm des Clients 39
5.2. Klassendiagramm für das Domain Model des Clients 43
5.3. Klassendiagramm für die Controller-Klassen des Clients 46
5.4. Klassendiagramm für die View-Methoden des Clients 47
5.5. Beispiel für eine YARA-Regel aus dem Client (hier zum Finden von Indikatoren für die Verwendung der Malware „DealersChoice“ durch APT28) 49
5.6. Beispiel für ein Objekt der Klasse „MatchDict“, wie abgebildet in der MongoDB-Instanz (Screenshot aus dem Tool „MongoDB Compass“[Monb]) 51

6.1. Deploymentdiagramm für die lokale Ausführung des Prototyps als containerisierte Webanwendung 57
6.2. Übersicht über Ordnerstruktur des Quellcodes (Screenshot aus PyCharm-IDE) 61
6.3. Inhalt der Ordner „mongo“ und „nginx“ des Quellcodes für die gleichnamigen Services (Screenshot aus PyCharm-IDE) 61

6.4. Inhalt des Ordners „webapp“ des Quellcodes für den gleichnamigen Service (Screenshot aus PyCharm-IDE) 62
6.5. Inhalt des Ordners „app“ des Quellcodes als Unterordner des „webapp“-Ordners (Screenshot aus PyCharm-IDE) 62
6.6. Beispiel für eine View Methode 63
6.7. Inhalt der Ordner „controllers“ und „classes“ des Quellcodes als Unterordner des „webapp“-Ordners (Screenshot aus PyCharm-IDE) 63
6.8. Inhalt der Ordner „mitre“ und „yara_rules“ des Quellcodes als Unterordner des „webapp“-Ordners (Screenshot aus PyCharm-IDE) 64
6.9. Navigationsleiste der Anwendung für einen eingeloggten User . . 64
6.10. Erster Teil der Falldarstellung: Auswertungsergebnis 65
6.11. Zweiter Teil der Falldarstellung: Mögliche Gegenmaßnahmen zu identifizierten Angriffen 65
6.12. Fehlermeldung bei Eingabe eines unerlaubten Fallnamens (dieser darf aus Sicherheitsgründen keine Sonderzeichen enthalten) . . . 66

7.1. Übersicht über identifizierte Techniken aus Test 1 70
7.2. Übersicht über identifizierte Techniken aus Test 2 71
7.3. Übersicht über identifizierte Techniken aus Test 3 73

Tabellenverzeichnis

4.1. Use Case U1: Create Account 30
4.2. Use Case U2: Log In 31
4.3. Use Case U3: Create Case 32
4.4. Use Case U4: Read Case Documentation 33
4.5. Use Case U5: Submit Log for Analysis 34
4.6. Use Case U6: Evaluate Mitigations 35

1. Einleitung

In den letzten Jahren ist die globale Cybersicherheitslandschaft zunehmenden Bedrohungen ausgesetzt. So machten sich etwa während der Covid-19-Pandemie Cyberkriminelle falsch eingerichtete Netzwerke zunutze, da Unternehmen in Remote-Arbeitsumgebungen auslagern mussten. Im Jahr 2020 nahmen Malware-Angriffe im Vergleich zu 2019 um 358% zu. Von hier aus stiegen Cyberangriffe bis 2021 weltweit um 125%, und auch im Jahr 2022 bedrohten weiterhin zunehmende Mengen an Cyberangriffen Unternehmen und Privatpersonen.[Cha] Cyberangriffe werden dabei tendenziell immer gefährlicher und setzen höher entwickelte Angriffsformen ein. Die Spitze dieser Angriffe bilden dabei Advanced Persistent Threats (APT) ab. Hierbei definiert das deutsche Bundesamt für Sicherheit in der Informationstechnik (BSI) einen APT folgendermaßen: *„Ein APT liegt dann vor, wenn ein gut ausgebildeter, typischerweise staatlich gesteuerter, Angreifender zum Zweck der Spionage oder Sabotage über einen längeren Zeitraum hinweg sehr gezielt ein Netz oder System angreift, sich unter Umständen darin bewegt und/oder ausbreitet und so Informationen sammelt oder Manipulationen vornimmt.“*[Bun]

1.1. Motivation und Ausgangssituation

Cybersicherheit ist sowohl in der Industrie als auch in der Forschung ein wichtiges Thema. Vielleicht gab es vor über einem Jahrzehnt eine Zeit, in der Cybersicherheit nur eine Frage des „Ob“ war; „Ob“ eine Organisation kompromittiert werden würde. Doch mittlerweile wurde es zu einer Frage „Wann“ und „auf welcher Ebene“. Trotz der Verbreitung von Cyberangriffsfähigkeiten und ihren potentiellen Auswirkungen schneiden viele Unternehmen in Bezug auf das Cybersicherheitsmanagement nach wie vor schlecht ab. Diese Unternehmen unterschätzen oder ignorieren Cyber-Risiken oder verlassen sich ausschließlich auf generische Standard-Cybersicherheitslösungen.[JSM19, S.1f] Diese Aussage wurde kürzlich erst von der österreichischen Tageszeitung „Der Standard“ untermauert:

„Dass das Internet Gefahren birgt, ist mittlerweile so gut wie allen bewusst. Geht es darum, präventiv etwas gegen diese Gefahren zu unternehmen, besteht in Österreich jedoch noch viel Aufholbedarf. Jedes dritte Unternehmen erkennt IT-Sicherheitsvorfälle nicht einmal, wie aus einer Analyse des KSV 1870 Nimbusec hervorgeht. Dabei ist das Thema allgegenwärtig, in den vergangenen Wochen

traf es mit Magenta, Rosenbauer und der GIS beispielsweise drei prominente Namen. Dennoch haben zahlreiche Unternehmen nach wie vor kaum bis keine Sicherheitsvorkehrungen getroffen oder überschätzen die eigenen Maßnahmen. Vor allem Klein- und Mittelbetriebe hätten großen Aufholbedarf, heißt es in der aktuellen Analyse.“ [And]

Essentielle Wirtschaftszweige, wie die globalen Lieferketten, sind ebenfalls stark betroffen, wie eine aktuelle Einschätzung des Beratungsunternehmens KPMG zeigt: Cyberkriminelle werden im Jahr 2023 wahrscheinlich noch gewandter sein, wenn es darum geht, Lieferketten zu infiltrieren, um Unternehmen zu berauben oder zu schädigen.[KPM] Wie kritisch die Situation bereits geworden ist und dass großangelegte Cyberangriffe heutzutage keine Seltenheit mehr sind, stellt darüber hinaus der folgende Beitrag unter Beweis:

„Die Hackergruppe „noname057(16)“ griff unter anderem die IT-Systeme der [italienischen] Regierung und des Parlaments, des Verkehrs- und des Außenministeriums, der Verkehrsregulierungsbehörde und der römischen Nahverkehrsgesellschaft ATAC an. Angegriffen wurde auch das IT-System des Flughafens Bologna, berichteten italienische Medien.“[ORF]

Bereits seit der Existenz von IT-Systemen sind die Sicherheit von Systemen und Daten sowie deren Schutz vor Angriffen und anderen Gefährdungen ein Thema. Für moderne Unternehmen ist Cybersicherheit heute eine wesentliche Anforderung. Allerdings sehen sich viele Unternehmen mit erheblichen Einschränkungen konfrontiert, da es an qualifiziertem Personal mangelt, um die erforderlichen Rollen und Verantwortlichkeiten zu erfüllen.[Fur21, S.1] Darüber hinaus können Cyberangriffe zu einer Situation führen, in dem Entscheidungsträger während Cyberangriffen nicht nur mit der Herausforderung mangelnder Informationen konfrontiert werden, sondern auch ihr Vertrauen in die verfügbaren Informationen verlieren, wenn sie angegriffen werden.[CG11, S.2632]

1.2. Problemstellung

Wie die bisherige Einleitung zeigt, treten Cyberangriffe häufiger auf und werden gefährlicher. Sie bedrohen zunehmend die kritische Infrastruktur und ganze Wirtschaftszweige. Um eine gewisse Abhilfe ermöglichen zu können, braucht es Methoden, um Cyberangriffe erkennen bzw. einordnen zu können. Hierzu sieht das National Institute of Standards and Technology (NIST) Framework bestehend aus seinen fünf Kernfunktionen (Identify, Protect, Detect, Respond und Recover) die Kernfunktion Detect vor. Im Rahmen von Detect werden geeignete Aktivitäten sowohl entwickelt als auch implementiert, um das Auftreten eines Cybersicherheitsereignisses zu erkennen.[oST18, S.7] In dieser Detect-Phase ist es von zentraler Bedeutung, die Umgebung, in der sich eine betroffene Organisation

befindet, zu untersuchen, um möglichst frühzeitig eine Bedrohung entdecken und Handlungsmaßnahmen ableiten zu können. Darüber hinaus ist es wichtig, zu wissen, was die nächsten Schritte sind und wie die Abwehr des Angriffs gelingen kann. Nachdem Ressourcen beschränkt sind, muss das Unternehmensumfeld fokussiert – ähnlich zu einem Lichtstrahl einer Taschenlampe – durchleuchtet werden. Für diese Beobachtung bietet sich MITRE ATT&CK an. Sie ist eine weltweit zugängliche Wissensdatenbank über feindliche Taktiken und Techniken, die auf Observierungen aus der realen Welt basiert.[Mit] Die Angriffe und dazugehörigen Gegenmaßnahmen sind in dieser Wissensdatenbank gut dokumentiert.

Doch die aktuellen Entscheidungsprobleme sind komplexer als in der Vergangenheit und erfordern Entscheidungsunterstützung. Die meisten realen Entscheidungssituationen sind einer begrenzten Rationalität ausgesetzt.[RGSM18, S.19] In der Cybersicherheit erfolgt die Entscheidungsfindung meist ad hoc und ist in hohem Maße von statischen Richtlinien sowie menschlichem Eingreifen abhängig.[HAC+16, S.1] Jedoch können falsche Entscheidungen, sowohl durch Sicherheitsvorfälle als auch durch Fehlinvestitionen und verpasste Chancen, bedeutsame negative Auswirkungen haben.[fSidI17, S.6] Der zunehmende Einsatz von Entscheidungsunterstützungssystemen (Decision Support Systems - DSS) in Organisationen in den letzten Jahrzehnten ist ein starker Nachweis dafür, dass DSS ein praktikables und gut akzeptiertes Managementinstrument darstellt.[RGSM18, S.20] Im Bereich der Cyber Security bzw. in erstklassigen Security Operations Center (SOC) sind im Bereich der DSS sogenannte Security Information and Event Management (SIEM) Systeme im Einsatz, wie Mitre in ihrem Buch „11 Strategies of a World-Class Cybersecurity Operations Center“ schildert. SIEM Systeme sammeln, aggregieren, filtern, speichern, selektieren, korrelieren und zeigen sicherheitsrelevante Daten an und unterstützen sowohl Echtzeit- als auch historische Überprüfungen und Analysen. Allerdings können SIEMs sowohl in ihrer Anschaffung als auch im laufenden Betrieb sehr teuer sein.[KK22, S.243] Darüber hinaus fällt es einigen SOCs schwer, den Mehrwert von SIEMs zu realisieren, was zum großen Teil auf ihre Komplexität zurückzuführen ist, da die effektive Erstellung und Pflege von Korrelationsregeln ressourcenintensiv sein kann.[KK22, S.244] Vorliegende Masterarbeit möchte hierbei ein high-level SIEM als DSS entwickeln, das in seiner Komplexität auf Basis des Masterarbeit-Rahmens auf den Kernbereich der Netzwerkaktivität fokussiert ist, um einen niederschwelligeren DSS-Prototypen anzubieten, als kommerziell verfügbare SIEMs. Für ein high-level SIEM, dass diesen Anforderungen gerecht wird, braucht es jedoch ein Entscheidungsmodell, anhand dessen eine Organisation mit dem DSS (inter)agieren kann.

Entscheidungsmodelle erfreuen sich in der Wirtschaft großer Beliebtheit.[Ros14, S.1] Im Bereich der Cyber Security hat sich der OODA Loop als Entscheidungsmodell bewährt, um digitalen Bedrohungen begegnen zu können.[MFM+19, S.1] Um das oben erwähnte NIST Framework (in der Detect Phase) durch ein fort-

schrittliches SOC im Rahmen eines SIEMs möglichst effektiv einsetzen zu können, braucht es ein ausgezeichnetes Entscheidungsmodell wie der erwähnte OODA Loop bzw. möglicherweise sogar eine Integration des OODA Loop Modells mit anderen bewährten Entscheidungsmodellen aus der Betriebswirtschaftslehre (BWL).

Vor dem Hintergrund, dass Entscheidungen komplex und mangelnde Informationen bei einem Cyber-Angriff herausfordernd sind und zu Fehlentscheidungen führen können, möchte die vorliegende Masterarbeit einen Beitrag leisten, Organisation bei der Identifikation von Cyber-Angriffen zu unterstützen. Um dies zu erreichen, behandelt diese Masterarbeit einerseits die Literaturanalyse zu bekannten betriebswirtschaftlichen Entscheidungsmodellen, mit dem Ziel diese Prozessmodelle zu einem integrierten Entscheidungsmodell zusammenzufassen, wodurch eine Ergänzung zum bewährten OODA Loop Entscheidungsmodell geboten wird. Andererseits widmet sich die Masterarbeit der Realisierung dieses Entscheidungsmodells anhand eines Client-Prototyps. Dieser Prototyp stellt ein Decision Support Tool dar, womit menschlichen Entscheidungsträgern geholfen werden soll, effektivere Information Security Entscheidungen treffen zu können. Das zu erstellende Entscheidungsmodell bzw. der Client-Prototyp richtet sich an Verantwortliche für die Informationssicherheit, Sicherheitsbeauftragte, -experten, und -berater (ähnlich dem Adressatenkreis des BSI-Standard 200-1).[fSidI17, S.6] Dabei liegt der Fokus auf der Analyse von den eingangs erwähnten APTs. Sie gehören einer Kategorie der Cyberkriminalität an, welche sich gegen geschäftliche und politische Ziele richtet. Um erfolgreich zu sein, erfordern APTs ein hohes Maß an Tarnung über eine längere Betriebsdauer. Die Angriffsziele gehen in der Regel über den unmittelbaren finanziellen Gewinn hinaus, und kompromittierte Systeme sind auch weiterhin funktionsfähig, obwohl wichtige Bestandteile angegriffen und die ursprünglichen Ziele erreicht wurden.[GP14, S.51] Die vom Prototyp durchzuführenden Analysen von APT Cyberangriffen konzentrieren sich auf das Microsoft Windows Betriebssystem, da es nach wie vor das bedeutendste Betriebssystem im Desktop Bereich ist: Microsoft Windows war im Januar 2023 mit einem Anteil von knapp über 74 Prozent das weltweit dominierende Desktop-Betriebssystem. Das Mac-Betriebssystem von Apple hat im Laufe der Jahre Marktanteile gewonnen, bleibt aber ein kleiner Akteur auf dem Markt für Desktop-Betriebssysteme. Linux, das dritt beliebteste Desktop-Betriebssystem, hat einen kleinen, aber stabilen Marktanteil, wie Abbildung 1.1 zeigt.[sta]

Die zentrale Fragestellung der vorliegenden Masterarbeit lautet: *„Wie können betriebswirtschaftliche Entscheidungsmodelle zur Unterstützung des Cyber Security Managements angewendet werden und bringen sie eine wesentliche Verbesserung zur Unterstützung des Incident Handling?“*

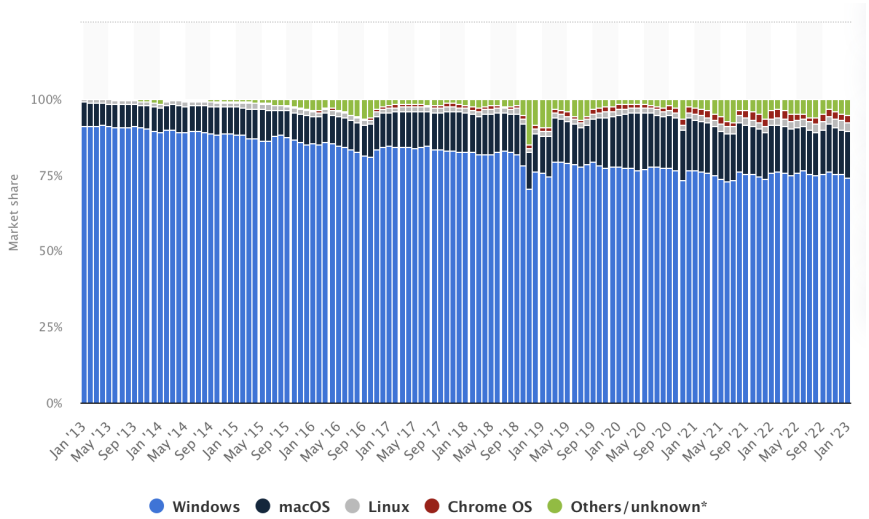


Abbildung 1.1.: Marktanteil von Betriebssystemen bei Desktop-PCs 2013–2023, nach Monat [sta].

1.3. Struktur der Masterarbeit

Nachdem in diesem Kapitel die Einleitung inkl. der Motivation und Ausgangssituation bzw. der Problemstellung und zentralen Fragestellung behandelt wurde, widmet sich das zweite Kapitel den theoretischen Grundlagen bzgl. der Entscheidungstheorie und Entscheidungsmodellen. Hier wird auf die Grundlagen der Entscheidungstheorie eingegangen, ein Exkurs zu Decision Support Systemen gemacht sowie die organisatorische Entscheidungsfindung thematisiert. Das Kapitel beinhaltet darüber hinaus die Vorstellung diverser Entscheidungsmodelle, die in späterer Folge als Grundlage für die Erstellung des integrierten Entscheidungsmodells fungieren. Im dritten Kapitel werden die jeweiligen Bestandteile der zuvor vorgestellten Entscheidungsmodelle benannt, die daraufhin verwendet werden, um das integrierte Entscheidungsmodell zu erstellen. Im Anschluss wird dieses Entscheidungsmodell veranschaulicht und beschrieben. Das vierte Kapitel beschäftigt sich mit den Beschreibungen der Use Cases und Requirements des Prototyps. Im fünften Kapitel wird detailliert das Design des Prototyps behandelt. So wird sowohl auf die geplante Nutzung des Clients als auch dessen Grundstruktur eingegangen. Des Weiteren wird auf das Hochladen und Auswerten von Logdateien eingegangen sowie die Darstellung der Analyseergebnisse und Feedback durchgenommen. Kapitel sechs behandelt die Implementierung des Prototyps und geht im Detail auf den verwendeten Technologiestack ein. Darüber hinaus wird die Architektur und das Deployment sowie die Ausführung des Clients beschrieben. Der Aufbau und die Struktur des Quellcodes, ebenso wie die Benutzeroberfläche und die Absicherung der Anwendung werden behandelt. Das Kapitel sieben widmet sich der Evaluierung, sprich des Tests des Prototyps. Hier wird auf die jeweiligen Testszenarien eingegangen, generelle Resultate der

Evaluierung werden geschildert und Empfehlungen aus der Evaluierung werden gegeben. Im achten und letzten Kapitel, dem Conclusio, wird auf mögliche Weiterentwicklung des Prototyps eingegangen sowie den zentralen Schlussfolgerungen, in denen die Beantwortung der zentralen Fragestellung vorgenommen wird.

2. Entscheidungstheorie und Entscheidungsmodelle

Dieses Kapitel beschäftigt sich mit der Literaturanalyse zu den Grundlagen der Entscheidungstheorie, Decision Support Systeme sowie der organisatorischen Entscheidungsfindung. Weiters stellt es diverse Entscheidungsmodelle vor, die in Abschnitt 3 in einem integrierten Entscheidungsmodell zusammengefasst werden.

2.1. Grundlagen der Entscheidungstheorie

Nahezu alles, was Menschen erledigen bzw. unternehmen, benötigt Entscheidungen. Die Entscheidungstheorie will hierbei Hilfestellungen liefern, wie Menschen „vernünftige Entscheidungen“ treffen können. Darüber hinaus will sie erklären, wie reale Entscheidungen entstehen. Sie beschäftigt sich sowohl mit dem Entscheidungsverhalten von Individuen, als auch Gruppen und Organisationen. Dabei werden entscheidungstheoretische Untersuchungen zwischen deskriptiver und präskriptiver (bzw. normativer) Entscheidungstheorie differenziert.[Gab] Das Ziel der präskriptiven bzw. normativen Entscheidungstheorie ist es zu zeigen, wie Entscheidungen „rational“, also vernünftig, getroffen werden können. Hiermit versucht sie Empfehlungen zu geben, wie Individuen oder Gruppen Entscheidungen in unterschiedlichen Entscheidungssituationen treffen sollten. Grundprobleme der Alternativenauswahl, die repräsentativ für zahlreiche reale Entscheidungssituationen sind (wie etwa die Abwägung von Ertrag und Risiko in einer Entscheidungssituation bei Unsicherheit) werden im Rahmen der normativen Entscheidungstheorie untersucht.[Gab] Das reale Entscheidungsverhalten zu beschreiben und zu erklären, ist das Ziel der deskriptiven Entscheidungstheorie. Hierzu sollen aus Observationen Hypothesen über das Verhalten von Individuen und Gruppen im Entscheidungsprozess ausgearbeitet werden. Mit deren Hilfe kann, bei Kenntnis der jeweiligen konkreten Entscheidungssituation, das entsprechende Entscheidungsverhalten prognostiziert werden.[Gab]

Charakterisierung der Teilgebiete

- **Präskriptive bzw. normative Entscheidungstheorie**
  Die normative bzw. präskriptive Entscheidungstheorie beschreibt nicht die Realität, sprich wie eine Entscheidung tatsächlich getroffen wird. Stattdessen gibt sie Empfehlungen über das Verhalten für alternative Entschei-

dungssituationen in der Realität. Diese Empfehlungen erfolgen, indem sie aufzeigt, wie Entscheidungen getroffen werden sollten, was einem deduktiven Vorgehen entspricht. Die normative Orientierung meint, dass ein Entscheider erstens konsequent im Einklang mit seinem Zielsystem agiert und zweitens die ihm zur Verfügung stehenden Informationen korrekt verarbeitet sowie weitere Informationen rational (sprich unter Abwägung ihrer Nutzen und Kosten) besorgt. Die Entwicklung von Entscheidungsmodellen ist unter anderem ein zentraler Bestandteil der präskriptiven Entscheidungstheorie. Hierbei wird ein Entscheidungsproblem in abstrakter Form und in einer formalisierten Sprache so dargestellt, dass die Lösung des Entscheidungsproblems logisch abgeleitet werden kann.[Gab]

- **Deskriptive Entscheidungstheorie**
  Die deskriptiven Theorien werden im Gegenzug zu den präskriptiven Entscheidungstheorien nicht deduktiv, sondern induktiv aus empirischen Observationen abgeleitet. Diese Beobachtungen werden aus dem Verhalten von Individuen bzw. Gruppen in realen Situationen und zusätzlichen aus kontrollierten Experimenten bezogen. Dabei stellt sich heraus, dass reale Entscheider dem Rationalitätsideal der normativen/präskriptiven Theorien kaum voll entsprechen. Die Annahme einer vollkommenen Rationalität menschlicher Entscheidungen wird daher von deskriptiven Entscheidungstheorien verworfen. Stattdessen bezieht sie - im Einklang mit psychologischen und soziologischen Erkenntnissen - die zahlreichen individuellen und sozialen Begrenzungsfaktoren der menschlichen Vernunft bzw. Rationalität in die Analyse mit ein. Die beachteten Begrenzungsfaktoren sind meistens kognitiver Art (z.B. die beschränkte Kapazität des Menschen zur Informationsverarbeitung).[Gab]

In Anbetracht dieser beiden Stoßrichtungen der Entscheidungstheorie und mit Hinblick auf die zugrundeliegende Masterarbeit, die sich mit der Erstellung eines Decision Support Systems befasst, liegt das zu beachtende theoretische Fundament im Bereich der normativen/präskriptiven Entscheidungstheorie, da das Decision Support System klare Handlungsanweisungen empfiehlt und sich nicht auf die Beschreibung, wie eine Entscheidung getroffen wird, fokussiert. Darüber hinaus ist die (mathematische) logische Lösung des Entscheidungsproblems nicht Gegenstand der vorliegenden Arbeit, da ein Decision Support System, auch ohne einem mathematischen Verfahren, Entscheidungen über eine qualitative Pro- und Kontra-Analyse ermöglicht.[RGSM18, S.27]

2.2. Decision Support Systeme

Managemententscheidungen leiten sich im Allgemeinen von menschlichem Urteilsvermögen ab, das durch Erfahrung, Informationen und Wissen unterstützt wird.[FE09, S.31] Um die Auswirkungen menschlicher Fehler zu kompensieren,

kann der Entscheidungsprozess teilweise durch computergestützte Automatisierung bereichert werden.[RGSM18, S.20] Ein Decision Support Systeme (DSS) ist eine Computerprogrammanwendung, die verwendet wird, um die Entscheidungsfähigkeit eines Unternehmens zu verbessern. Es analysiert große Datenmengen und präsentiert einer Organisation die bestmöglichen verfügbaren Optionen. Ein typisches DSS besteht aus drei verschiedenen Teilen: Knowledge Base (Wissensdatenbank), Software und Benutzeroberfläche. Es gibt verschiedene Arten von DSS, welche in Kategorien unterteilt werden können, die jeweils auf ihren primären Informationsquellen basieren: Datengesteuertes-, Modellgesteuertes-, Kommunikationsgesteuertes- bzw. Gruppen-, Wissensgesteuertes- und Dokumentengesteuertes-DSS.[Tecb] Das DSS der vorliegenden Masterarbeit zählt zur Kategorie des Wissensgesteuertem-DSS, denn bei dieser Art von DSS befinden sich die Daten, die das System antreiben, in einer Wissensbasis, die kontinuierlich aktualisiert wird. Ein Wissensgesteuertes-DSS stellt Benutzern Informationen bereit, die mit dem Wissensstand eines Unternehmens übereinstimmen.[Tecb]

Die idealen Eigenschaften und Fähigkeiten eines DSS werden unter anderem wie folgt zusammengefasst:[RGSM18, S.20f]

- DSS unterstützen Entscheidungsträger bei halbstrukturierten und unstrukturierten Problemen, wobei menschliches Urteilsvermögen und Computer eingesetzt werden.
- DSS decken ein breites Spektrum an Führungsebenen ab, vom Top-Executive bis zum Linienmanager.
- DSS unterstützen sowohl Einzelpersonen als auch Gruppen. Weniger strukturierte Situationen erfordern oft das Eingreifen mehrerer Personen aus verschiedenen Bereichen und Organisationsebenen oder manchmal sogar aus verschiedenen Organisationen.
- DSS erleichtern mehrere voneinander abhängige und/oder aufeinanderfolgende Entscheidungen, die einmal, mehrmals oder wiederholt getroffen werden können.
- DSS versuchen eher die Effektivität der Entscheidungsfindung (Angemessenheit und Qualität) als ihre Effizienz (Kosten der Entscheidungsfindung) zu verbessern.
- DSS versuchen die Entscheidungsträger zu unterstützen, nicht sie zu ersetzen. Daher haben sie die Kontrolle über alle Ebenen des Prozesses.

2.3. Organisatorische Entscheidungsfindung

Nachdem das Decision Support Tool in einem organisatorischen Umfeld zur Anwendung gelangen soll, behandelt dieses Unterkapitel das theoretische Fundament

der organisatorische Entscheidungsfindung sowie Arten der organisatorischen Entscheidungsfindung.

Organisatorische Entscheidungsfindung

wird von Jones definiert, als den „Prozess der Reaktion auf ein Problem durch Suche und Auswahl einer Lösung oder Vorgehensweise, die den größten Nutzen für die organisatorischen Stakeholder schafft".[Jon13, S.334] Daft definiert organisatorische Entscheidungen in ähnlicher Weise als „den Prozess der Identifizierung und Lösung von Problemen" und erweitert seine Definition, da er genaueres Augenmerk auf zwei Phasen beim Entscheidungsfindungsprozess legt: Problemidentifikation und Problemlösung. In der Phase der Problemidentifizierung werden Informationen über die äußeren und inneren Umgebungsbedingungen einer Organisation überwacht, um festzustellen, ob die Leistung der Organisation zufriedenstellend ist und um die Ursache etwaiger Mängel zu diagnostizieren. In der Problemlösungsphase werden alternative Vorgehensweisen berücksichtigt und die beste Alternative ausgewählt und durchgeführt.[Daf13, S.478] Nach Lynch [Lyn11] ist die Entscheidungsfindung, im Allgemeinen, eine organisatorische Funktion, die jeder Aktion vorausgeht. Diese Entscheidungen können grob als operativ, taktisch oder strategisch eingestuft werden.[KVBAO16, S.2] Simon [Sim60] stellte fest, dass Entscheidungen, die innerhalb von Organisationen getroffen werden, in ihrer Komplexität variieren und daher am besten als programmiert oder nicht-programmiert kategorisiert werden können.[LS14, S.14]

Programmierte Entscheidungen

befassen sich in der Regel mit Situationen, die genau definiert sind und/oder sich wiederholen. Schoemaker & Russo [SR93] und Daft [Daf13] legen nahe, dass standardisierte Organisationsabläufe unter diesen Umständen gut funktionieren können, da die Leistungskriterien klar definiert sind, die Informationsverfügbarkeit gut ist, die Entscheidungsalternativen leicht spezifiziert werden können sowie eine hohe Erfolgswahrscheinlichkeit bei Auswahl einer der bekannten Alternativen gewährleistet ist.[LS14, S.14]

Nicht-Programmierte Entscheidungen

tendieren lt. Daft [Daf13] dazu, neuartig und schlecht definiert zu sein, ohne dass sie etablierte Verfahren zur Lösung des Problems anbieten. Darüber hinaus ist es lt. Jones [Jon13] praktisch unmöglich, Regeln, Routinen oder Standardarbeitsanweisungen im Voraus festzustellen, da die Situationen einzigartig und/oder unerwartet sind.[LS14, S.14] Weiters hält Simon [Sim60] fest, wenn eine Organisation ein bestimmtes Problem noch nie zuvor gesehen hat, weiß sie möglicherweise nicht, wie sie reagieren soll. Infolgedessen müssen Lösungen oft erst gefunden werden, nachdem die neuen Probleme aufgetreten sind.[LS14, S.14] Daft [Daf13] ergänzt, dass es bei nicht programmierten Entscheidungen häufig

keine eindeutigen Entscheidungskriterien gibt. Außerdem sind die Alternativen in der Regel unscharf, und es besteht ein gewisses Maß an Unsicherheit darüber, ob die vorgeschlagene Lösung das Problem tatsächlich lösen wird.[LS14, S.14] Infolgedessen erfordern nicht-programmierte Entscheidungen lt. Jones [Jon13] ein höheres Maß an Informationssuche und der Notwendigkeit stärkerer Zusammenarbeit zwischen Managern, Funktionen und Abteilungen als mit einer programmierten Entscheidung. Aufgrund ihrer inhärenten Komplexität zwingen nicht-programmierte Entscheidungen Manager Urteilsvermögen, Intuition und Kreativität einzusetzen, um diese Probleme zu lösen.[LS14, S.14f] Einige nicht-programmierte Entscheidungen führen zu Standardarbeitsanweisungen, um das Problem zu lösen, sollte die Situation erneut auftreten. Hierdurch wird aus einer nicht-programmierten Entscheidung eine programmierte Entscheidung. Andere nicht-programmierte Entscheidungen wie die strategische Planung bleiben einzigartig und können nicht durch programmierte Entscheidungen abgelöst werden.[LS14, S.15]

Rashidi et al. erwähnen, dass Management-Entscheidungen im Allgemeinen anhand verfügbarer Daten und Informationen getroffen werden, die von Natur aus meist vage, ungenau und ungewiss sind. Wobei Multi-attribute-decision-making (MADM) ein effizientes Werkzeug für den Umgang mit Unsicherheiten ist.[RGSM18, S.26] MADM enthält elementare Ansätze, welche sich durch ihre Einfachheit und ihre Unabhängigkeit von rechnerischer Unterstützung auszeichnen. Sie eignen sich für Probleme mit einem einzelnen Entscheidungsträger sowie begrenzter Alternativen und Kriterien. Maximin- und Maximax-Methoden, Pro- und Kontra-Analyse, konjunktive und disjunktive Methoden und die lexikografische Methode fallen alle in diese Kategorie.[RGSM18, S.27] Nachdem der Fokus der vorliegenden Masterarbeit nicht auf mathematischen Analysen liegt, kann ein Entscheidungsträger für den praktischen Einsatz des implementieren Prototypen die Pro- und Kontra-Analyse (bzw. Vor- und Nachteil-Analyse) verwenden. Sie ist eine qualitative Vergleichsmethode, bei der positive und negative Aspekte jeder Alternative bewertet und verglichen werden. Diese Methode ist einfach in der Praxis umzusetzen, da keine mathematischen Kenntnisse erforderlich sind.[RGSM18, S.27]

2.4. Entscheidungsmodelle

In diesem Unterkapitel werden einige bekannte Entscheidungsmodelle aus der Literatur vorgestellt. Das weiterführende Ziel dieses Unterkapitels ist es, nach der Vorstellung der Modelle, gewisse Bestandteile der Modelle auszuwählen, die in ein integriertes Entscheidungsmodell überführt werden (siehe dazu Abschnitt 3).

Unabhängig von den nun folgenden Modellen hat die Entscheidungsfindung grundsätzlich zwei verwandte Aspekte. Der erste ist eine Abfolge bzw. Sequenz

von Schritten oder ein Prozess, der von einer Organisation verwendet wird, um eine Lösung zu erreichen. Der zweite Aspekt bezieht sich darauf, wie Lösungen identifiziert und überprüft werden, wenn eine Entscheidung innerhalb einer Organisation getroffen wird.[LS14, S.18] Die organisatorische Entscheidungsfindung wurde lt. Kiesler [KS82, S.548] traditionell als rationaler Prozess dargestellt, in denen allwissende Manager Entscheidungen treffen, die es Organisationen ermöglichen, sich auf ihre Umgebung einzustellen. Mittlerweile ist lt. Jones [Jon13] jedoch bekannt, dass der Entscheidungsprozess für Manager weitaus komplizierter und von Natur aus unsicher ist.[LS14, S.21]

Auf den folgenden Seiten werden eine Reihe von organisatorischen Entscheidungsmodellen beschrieben, die Entscheidungsträgern zur Verfügung stehen, um den Prozess der Entscheidungsfindung leiten zu können. Diese Liste der Modelle ist nicht vollständig, gibt aber einen Überblick über die Arten von Entscheidungsprozessen, die in Organisationen verwendet werden:[LS14, S.22]

- Rational Model
- Bounded Rationality Model
- Carnegie Modell
- Incremental Decision Model
- OODA Loop
- Recognition Primed Decsion Model

Rational Model

Lt. Simon [Sim60] und Jones [Jon13] legt das rationale Modell nahe, dass die Entscheidungsfindung ein Prozess in drei Phasen ist.[LS14, S.22] In Phase 1 muss das Problem identifiziert und definiert werden. In Phase 2 werden Alternativen zur Lösung des Problems generiert. In Phase 3 wird eine Lösung ausgewählt und implementiert. Lee et al. stellt diese drei Phasen in einem erweiterten sieben Schritte-Modell da, wie Abbildung 2.1 zeigt.[LLPE10, S.77] Rizun geht einen Prozessschritt weiter und bildet das Entscheidungsmodell zusätzlich in vier Phasen - Intelligence, Design, Choice und Implementation - ab, wie Abbildung 2.2 zeigt.[Riz14, S.214] Nach Jones [Jon13] liegen dem rationalen Modell drei Annahmen zugrunde:[LS14, S.22]

1. Entscheidungsträger verfügen über alle Informationen, die sie benötigen;
2. Entscheidungsträger haben die Fähigkeit, die beste Entscheidungen zu treffen;
3. Entscheidungsträger sind sich einig, was zu tun ist.

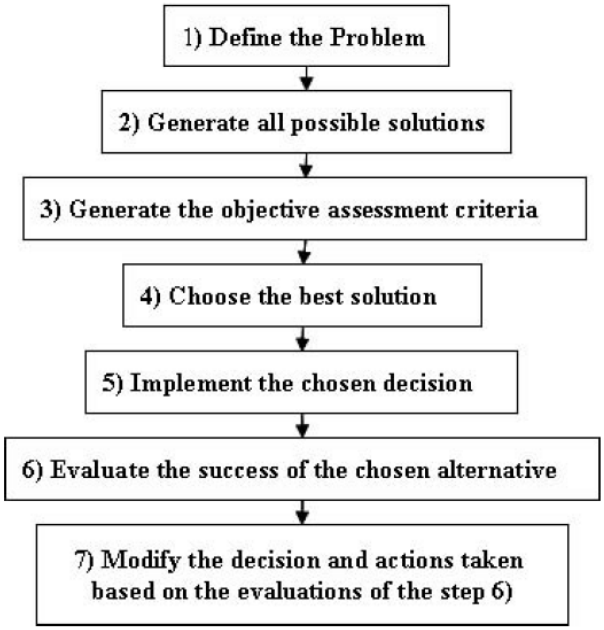


Abbildung 2.1.: Logische Prozesse mit sieben Schritten des RDM-Modells (Rational Decision Making) [LLPE10, S.77].

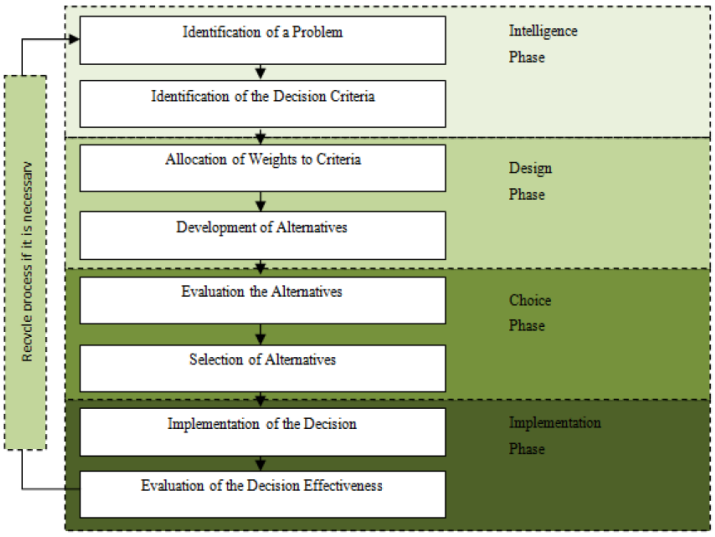


Abbildung 2.2.: Entscheidungsprozess des rationalen Modells [Riz14, S.214].

Diese Annahmen werden noch erweitert um Bedingungen der Gewissheit: alle Alternativen sind bekannt, alle Ergebnisse sind bekannt und alle Entscheidungskriterien sind bekannt.[Riz14, S.213]

Bounded Rationality Model

Wie anhand der zuvor erwähnten Annahmen dargelegt, hat das rationale Modell lt. Daft [Daf13] seine Grenzen: insbesondere wenn der Entscheidungsträger nicht alle Informationen zur Verfügung hat. Angesichts der Notwendigkeit, schnelle Entscheidungen mit unvollkommenen Informationen zu treffen, haben Entscheidungsträger nur eine bestimmte Zeit und geistige Kapazität, um jedes Problem zu bewerten und nach einer alternativen Lösung zu suchen. Infolgedessen sind die Versuche bei der Entscheidungsfindung rational zu sein begrenzt.[LS14, S.22] Die Perspektive der begrenzten Rationalität erkennt gewisse Einschränkungen an, die Entscheidungsträgern bzw. Managern auferlegt werden, die zu Kompromissen im Entscheidungsprozess führen können, wodurch keine optimalen Entscheidungen getroffen werden können.[LS14, S.23] Daft [Daf13, S.485] fasst die Einschränkungen, welche eine vollkommene Rationalität erschweren in folgende Punkte zusammen:[LS14, S.23]

- Zeitdruck
- Mangel an Informationen
- Ressourcenknappheit
- Organisatorische Einschränkungen (Übereinstimmung zwischen den Entscheidungsträgern, gemeinsame Perspektiven innerhalb der Organisation, Zusammenarbeit oder Unterstützung sowie die Unternehmenskultur, ethische Werte und Struktur)
- Persönliche Einschränkungen

Im Gegensatz zur vollständigen Rationalität impliziert das Bounded Rationality Model die folgenden Schritte nach Simon [Sim09], dargestellt in Abbildung 2.3 von Rizun. [Riz14, S.215]

Lt. Rizun sind die Grundideen der Prozessschritte des Bounded Rationality Model:[Riz14, S.215f]

1. Entscheidungen beruhen auf einem unvollständigen und unzureichenden Verständnis des Problems. Normalerweise bewirkt dies eine Reaktion in Form von einer Reduktion des Problems auf ein gewisses Niveau, auf dem das Problem wirklich verstanden werden kann.
2. Entscheidungsträgern gelingt es nie, alle möglichen alternativen Lösungen zu generieren. Sie konstruieren daher vereinfachte Modelle, die die wesentlichen Merkmale aus dem Problem extrahieren, ohne ihre gesamte Komplexität zu erfassen.

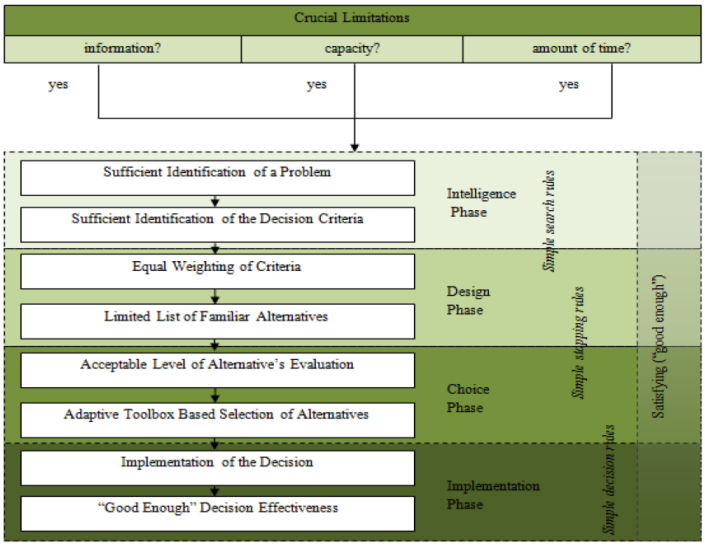


Abbildung 2.3.: Entscheidungsprozess des Bounded Rationality Models [Riz14, S.215].

3. Alternativen werden immer unvollständig bewertet. Auf bekannten Wegen prüft der Entscheider Alternativen nur so lange, bis er eine Alternative identifiziert, die „gut genug“ ist, sprich eine Alternative, die ein akzeptables Leistungsniveau erreicht.

4. Die endgültige Lösung ist eher eine zufriedenstellende als eine optimale Wahl.

Carnegie Model

Lt. Daft [Daf13] ging die Forschung bis zur Entwicklung des Carnegie Models davon aus, dass Entscheidungen von einer einzelnen Einheit innerhalb einer Organisation getroffen wurden, normalerweise des Geschäftsführers und dass alle relevanten Informationen an diesen Top-Entscheider weitergeleitet wurden, damit er oder sie eine Entscheidung trifft.[LS14, S.24] Lt. Stevenson et al. [SPP85] fanden die Forscher der Carnegie-Melon Universität jedoch heraus, dass Entscheidungen auf Organisationsebene oft viele Manager betrafen, und das die endgültige Wahl aus den Entscheidungsalternativen auf Koalitionen dieser Manager basierte. Diese Koalitionen werden als Allianzen zwischen Managern angesehen, die untereinander über die Ziele und Problemprioritäten der Organisation zugestimmt haben.[LS14, S.24] Lt. Daft [Daf13, S.491] werden Managementkoalitionen als notwendiger Bestandteil der Entscheidungsfindung aufgrund zweier Gründe gesehen: [LS14, S.24]

1. Organisatorische Ziele sind in der Regel nicht eindeutig und die operativen Ziele der jeweiligen Abteilung sind im gesamten Unternehmen inkonsistent.

2. Während Manager es vorziehen würden, völlig rationale Entscheidungen zu treffen, führen menschliche kognitive Einschränkungen, Zeitdruck und andere Beschränkungen dazu, dass Manager untereinander Informationen austauschen, um Mehrdeutigkeiten zu reduzieren, was letztendlich zum Aufbau von Koalitionen mit anderen Gleichgesinnten führt.

Incremental Decision Model

Die Forschung von Mintzberg et al. [MRT76] nahm einen anderen Blickwinkel im Vergleich zum Carnegie-Modell ein, wie Entscheidungen in Organisationen getroffen werden. Das Modell von Mintzberg et al. wird als inkrementelles Entscheidungsmodell bezeichnet und legt lt. Pienfield [Pin86] weniger Wert auf die politischen- und sozialen Aspekte des Koalitionsaufbaus im Carnegie-Modell, sondern konzentriert sich auf eine strukturierte Abfolge von Aktivitäten, die von der ersten Entdeckung des Problem bis zu seiner Lösung abzielt.[LS14, S.24f] Lt. Daft [Daf13] war ein Ergebnis der Mintzberg-Studie, das wichtige organisatorische Entscheidungen in der Regel aus einer Reihe kleiner Entscheidungen bestehen, die in Kombination die Hauptentscheidung ergibt. Während eine Organisation diese Reihe kleiner Entscheidungen durchläuft, kann es vorkommen, dass sie auf eine Barriere stößt, die sie am Vorankommen der Entscheidungsfindung hindert. Mintzberg et al. benannten diese Barrieren als „decision interrupts“ bzw. Entscheidungsunterbrechungen. Wenn eine Organisation auf eine Entscheidungsunterbrechung trifft, kehrt sie zur letzten getroffenen Entscheidung zurück und probiert dann etwas anderes aus. Mintzberg war der Meinung, dass dies ein wichtiger Teil des organisatorischen Lernens darstellt, da eine Organisation hiermit inkrementell lernt, was funktioniert und was nicht, wodurch sich das unternehmerische Risiko der Organisation letztlich reduziert. Eine weitere Implikation dieser Methodik ist, dass sich die endgültige Lösung für das Problem im Endeffekt sehr unterscheiden davon kann, was ursprünglich erwartet wurde.[LS14, S.25]

Aus ihrer Forschung entwickelten Mintzberg et al. [MRT76] ein Entscheidungsmodell zur Beschreibung der Reihenfolge der Entscheidungen, die getroffen werden, um zu einer wichtigen Entscheidung zu gelangen. Hierbei gibt es drei Hauptentscheidungsphasen: Identifizierung, Entwicklung und Auswahl. In der Identifizierungsphase wird das Problem zuerst erkannt (Recognition-Phase) und dann diagnostiziert (Diagnosis-Phase). In der Entwicklungsphase beginnt das Unternehmen zunächst, eine Lösung für das Problem zu finden, in dem es nach bereits bestehende Verfahren, die angewendet werden könnten, innerhalb des Unternehmens sucht (Search-Phase). Wenn das Unternehmen keine adäquate Lösungen für das Problem vorfindet, wird eine individuelle Lösung entwickelt (Design-Phase), durch den von Mintzberg et al. beschriebenen inkrementellen Ansatz. In der Auswahlphase werden schließlich Optionen ermittelt (Evaluation/Choice-Phase). Der Auswahlprozess erfordert, dass Manager/Führungskräfte bzw. Entscheidungsträger urteilen, analysieren und gegebenfalls

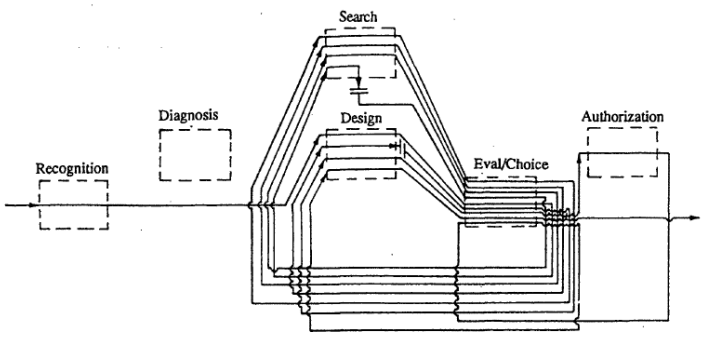


Abbildung 2.4.: Organisatorische Entscheidungsfindung als iterative Sequenz, motiviert durch Diagnose und unterbrochen durch Ereignisse [MRT76, S.273].

verhandeln, sofern eine Koalition benötigt wird (Authorization-Phase), ähnlich wie es im Carnegie-Modell beschrieben ist.[LS14, S.25]
Die eben erwähnten Schritte, werden in Abbildung 2.4 dargestellt.

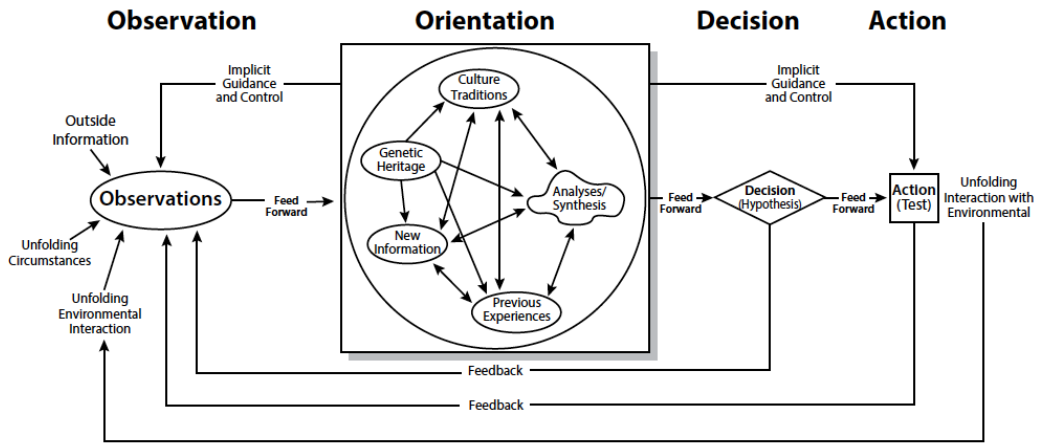
OODA Loop

Der OODA Loop (OODA-Schleife) von Oberst John Boyd ist ein weit verbreitetes Entscheidungsmodell[ZZ17, S.1] und in der Cyber Security Domain bewährt.[CG11, S.2637] Sie teilt sich in vier Phasen auf, wie Abbildung 2.5 zeigt: Erste Phase Observe (Beobachten), zweite Phase Orient (Orientieren), dritte Phase Decide (Entscheiden) und vierte Phase Act (Handeln).

Der OODA-Schleifenprozess ist ein nichtlinearer Prozess mit ständigem Feedback und Feed-Forward-Kanälen. Er bietet hierbei ein Analysewerkzeug und ein Modell, wie Entscheidungen getroffen werden sollen.[Boy18, S.385] Lt. Quirchmayr [Qui18, S.37] „...liegt der Vorteil des OODA-Loop-Modells darin, dass in den Phasen Observe und Orient explizit auf die Beschaffung von Informationen durch gezielte Beobachtung der Umgebung (Ereignisse und Verhaltensweisen, die auf Angreifer hindeuten) abgezielt wird.“ Die abstrakte Formulierung der OODA Loop von Boyd [Boy18, S.384f] verdeutlicht Quirchmayr am Beispiel der kooperativen Zielerreichung im Umfeld der Cybersecurity:

1. Phase: Observe - Diese Phase wird dazu benutzt, sich einen Überblick über die Situation zu verschaffen. Sämtliche Aktionen, die eine Bedrohung darstellen, werden beobachtet. Wenn diese Phase von einer äußeren Entwicklung losgetreten wird, muss diese Entwicklung ebenfalls beobachtet werden. Verhaltensregeln bzw. Einsatzregeln müssen für jeden Anlassfall überprüft werden. Sofern keine Einsatzregeln vorhanden sind, müssen diese erstellt werden.[Qui18, S.39]

2. Phase: Orient - In dieser Phase werden die Ergebnisse aus der ersten Phase



The OODA Loop. Boyd's final sketch of the OODA Loop, as presented in his summation of "A Discourse on Winning and Losing," which he referred to as "the big squeeze," 28 June 1995. Adapted from Hammond, *Mind of War, 190.*

Abbildung 2.5.: OODA-Loop [Boy18, S.384].

und die Einsatzregeln zusammengefasst sowie mögliche Vorgehensweisen definiert.[Qui18, S.39]

3. Phase: Decide - In dieser Phase befindet sich der Entscheidungsfindungsprozess.[Qui18, S.39]

4. Phase: Act - In dieser Phase werden die Aktionen gesetzt, die zum zuvor definierten Ziel führen. Im Anschluss muss Feedback erstellt werden und der Prozess startet von vorne. Sollten sich in dieser Phase neue Ereignisse entwickelt haben, die in der vorherigen Phasen nicht berücksichtigt wurden, muss der Prozess hier ohne eine gesetzte Aktion enden und mit den zusätzlichen Erkenntnissen in der ersten Phase erneut starten.[Qui18, S.46f]

Recognition Primed Decision Model

Eines der bekanntesten Entscheidungsmodelle ist Kleins [Kle98] Recognition-Primed Decision Making Model (RPD Model) bzw. auf Deutsch: Modell der erkennensbasierten Entscheidungsfindung.[NSR00, S.217] Studien über das Entscheidungsverhalten von Experten auf verschiedenen Gebieten zeigten, dass ein großer Teil ihrer Entscheidungen anhand des Modells von Klein getroffen werden.[Kle98, S.100] Das Wichtigste an diesem Modell ist, dass der Schwerpunkt auf der Situationsbewertung liegt. Sobald der Entscheider die Situation erkannt hat, gibt es vier Nebenprodukte (by-products):[NSR00, S.218]

- Expectancies - Der Entscheidungsträger erwartet, dass bestimmte Dinge geschehen, andere jedoch nicht;
- Relevant Cues - Der Entscheidungsträger achtet auf bestimmte Hinweise, um die Diagnose zu unterstützen;
- Plausible Goals - Es besteht ein gewisses Verständnis dafür, welche Ziele plausibel zu erreichen sind;
- Action 1...n - Es gibt bestimmte Aktionen, die wahrscheinlich erfolgreich sein werden.

Der Entscheidungsträger wählt eine Aktion, führt eine schnelle mentale Simulation der Aktion aus und wenn er glaubt, dass sie erfolgreich sein wird, setzt er die Aktion um. Wenn eine Aktion ausgewählt wurde, wird die Situation überwacht, um sicherzustellen, dass sie unverändert bleibt, wie erwartet. Sollte sich die Situation verändern, müssen möglicherweise alternative Aktionen in Betracht gezogen werden. Andernfalls berücksichtigt der Entscheidungsträger keine anderen Optionen. Im RPD Model wählt der Entscheidungsträger normalerweise eine Vorgehensweise „automatisch“ aus, sobald die Situation von ihm erkannt wird. Die gewählte Vorgehensweise ist wahrscheinlich eine, die zuvor bereits erfolgreich in dieser Situation war.[NSR00, S.218] Das Modell hat daher die zentrale Idee, dass eine Person, die Fachwissen besitzt, besser in der Lage ist, subtile Unterschiede zwischen Situationen zu erkennen und geeignete Vorgehensweisen entsprechend zu wählen.[NSR00, S.219]

Das RPD-Modell verbindet zwei Prozesse: Wie Entscheidungsträger die Situation einschätzen, um zu erkennen, welcher Handlungsverlauf Sinn macht, und die Art und Weise, wie sie diese Vorgehensweise bewerten, indem sie sich die Handlung vorstellen, bevor sie die Aktion gesetzt haben.[Kle98, S.24]

Abbildung 2.6 zeigt die Grundstrategie als Variante 1. Entscheidungsträger erkennen die Situation als typisch bzw. vertraut an und ergreifen eine Handlung bzw. Maßnahmen. Sie verstehen, welche Arten von Zielen Sinn machen (dadurch werden Prioritäten gesetzt), welche Hinweise wichtig sind (dadurch gibt es keine Informationsüberladung), was als nächstes zu erwarten ist (damit sie sich vorbereiten können und Überraschungen bemerken) und die typischen Wege, wie in einer bestimmten Situation zu reagieren ist. Indem man eine Situation als typisch begreift, erkennt man auch eine Vorgehensweise an, die wahrscheinlich erfolgreich sein wird. Die Erfassung von Zielen, Hinweisen, Erwartungen und Handlungen ist Bestandteil davon, eine Situation zu erkennen. Das bedeutet, dass die Entscheidungsträger nicht damit beginnen Ziele oder Erwartungen herauszufinden, sondern die Art der Situation.[Kle98, S.24]

Einige Situationen sind komplexer, wie die Varianten 2 und 3 in Abbildung 2.6 zeigen. Variante 2 tritt auf, wenn der Entscheidungsträger der Diagnose der Situation mehr Aufmerksamkeit widmen muss, da die Informationen möglicherweise nicht eindeutig übereinstimmen mit einem typischen Fall oder sie

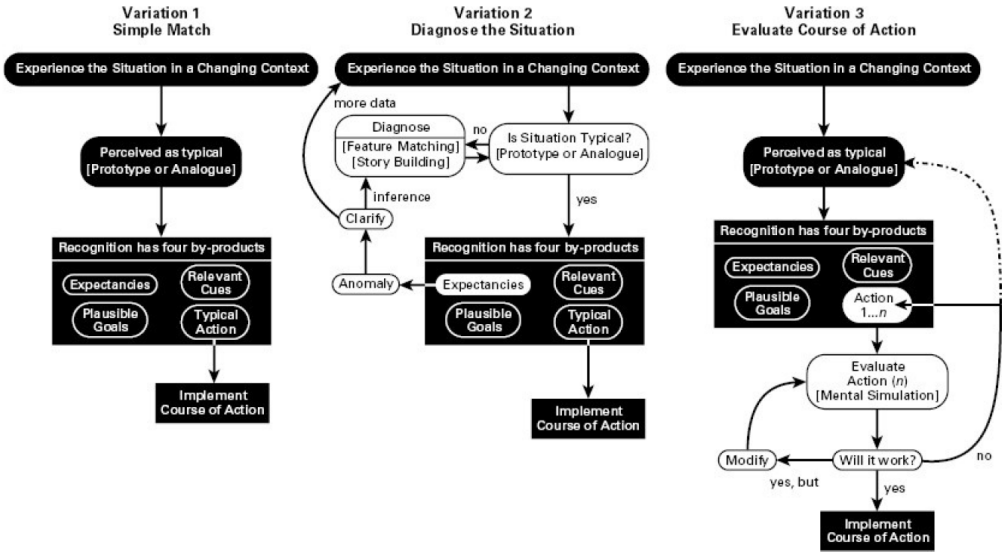


Abbildung 2.6.: Recognition Primed Decision Model [Kle98, S.25].

möglicherweise mehr als einem typischen Fall zugeordnet werden können. Der Entscheidungsträger muss möglicherweise weitere Informationen sammeln, um eine Diagnose zu stellen. Eine weitere Komplikation kann sein, dass der Entscheidungsträger die Situation möglicherweise falsch interpretiert hat und dies erst bemerkt, nachdem einige Erwartungen verletzt wurden. Wenn dies passiert, reagieren Entscheidungsträger auf die Anomalie oder Mehrdeutigkeit, indem sie überprüfen, welche Interpretation am besten zu den Merkmalen (Features) der Situation passen.[Kle98, S.25] Variante 3 erläutert, wie Entscheidungsträger einzelne Optionen bewerten, indem sie sich vorstellen, wie sich die gewählte Vorgehensweise entwickeln wird. Ein Entscheider, der Schwierigkeiten antizipiert, wird möglicherweise die Vorgehensweise anpassen bzw. sie ablehnen und nach einer anderen Option suchen.[Kle98, S.25]

Eine Möglichkeit, über diese drei Variationen nachzudenken, ist, dass die erste Variante im Grunde genommen eine „if... then“ Reaktion ist - eine Vorbedingung, gefolgt von einer regelbasierten Antwort. Das Fachwissen besteht darin, erkennen zu können, wann die Vorbedingung erfüllt wurde. Die zweite Variante hat die Form „if (???)... then“, wobei der Entscheidungsträger über die Art der Situation nachdenkt. Die dritte Variante hat die Form „if ... then (???)“, da der Entscheidungsträger über das Ergebnis einer Reaktion nachdenkt.[Kle98, S.25] Abbildung 2.7 zeigt eine integrierte Version aller drei Varianten.

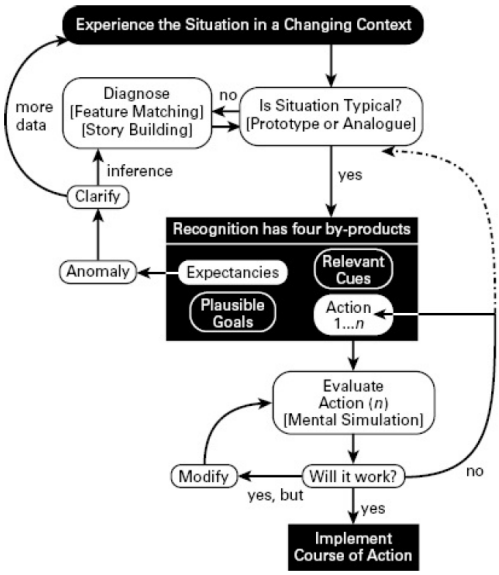


Abbildung 2.7.: Integrierte Version des Recognition Primed Decision Models [Kle98, S.26].

3. Integriertes Entscheidungsmodell

Nachdem die theoretischen Grundlagen behandelt wurden, werden in diesem Kapitel die jeweiligen Bestandteile der erwähnten Entscheidugsmodelle benannt, die für die Erstellung des integrierten Entscheidungsmodells herangezogen wurden. Im Anschluss wird das daraus abgeleitete Entscheidungsmodell vorgestellt.

3.1. Verwendung der vorgestellten Entscheidungsmodelle

Da aus den zuvor beschriebenen Entscheidungsmodellen ein integriertes Modell erstellt wird, ist eine zentrale Erkenntnis aus der bereits erwähnten Theorie, dass hierdurch ein normativer bzw. präskriptiver Ansatz im Rahmen der Entscheidungstheorie verfolgt wird. Darüber hinaus sollen sowohl nicht-programmierte wie auch programmierte Entscheidungen durch den Einsatz des integrierten Entscheidungsmodells möglich gemacht werden, um Organisationen dabei zu helfen, beide Arten von Entscheidungen im Bereich des Information Security treffen zu können. Um dies zu erreichen, werden bestimmte Bestandteile aus allen sechs behandelten Entscheidungsmodellen in die Erstellung des integrierten Entscheidungsmodells einfließen, da jedes Modell für sich wertvolle Erkenntnisse anbietet.

Das Rational Model und das Bounded Rationality Model werden hierbei als das theoretische Fundament herangezogen, was den generellen Ansatz der vollständigen bzw. unvollständigen Informationslage in einer Situation bzw. bei der Alternativenerstellung und -selektion Rechnung tragen soll. Hierzu wird der universelle Abfolgeprozess der Entscheidungsfindung verwendet, der sich grob in drei Phasen gliedert: 1. Problemidentifikation bzw. Problemdefinition, 2. Erstellung von Alternativen und Selektion, 3. Entscheidungsimplementierung.

Das Incremental Model findet Anwendung im integrierten Entscheidungsmodell, indem die jeweilige Evaluierung einer Alternativen Rechnung getragen werden soll. Darüber hinaus soll auf das Konzept der „decision interrupts“ (Entscheidungsunterbrechungen) eingegangen werden, wenn während des Durchlaufs des Entscheidungsprozesses ein erkannter Informationsmangel stattfindet. Schließlich soll noch die letzte Phase des inkrementellen Modells „Authorization“ am Ende des Entscheidungsprozess eingebunden werden.

Der OODA Loop wird im integrierten Modell Berücksichtigung finden auf Basis all seiner vier Phasen, „Observe“, „Orient“, „Decide“ und „Act“. Darüber hinaus wird auf sein Konzept der Feedbackschleifen zurückgegriffen, um organisatorisches Lernen zu ermöglichen.

Das Recognition Primed Decision Model wird anhand seiner Stärke der Situationsbewertung herangezogen. Das integrierte Entscheidungsmodell soll daher die Einschätzung berücksichtigen, ob eine Situation bereits bekannt oder neu ist. Hierdurch könnten Entscheidungsträger bereits frühzeitig bemerken, dass eine neue Situation eingetreten ist und daher mit der aktuellen Informationslage nur eine unsichere Entscheidung getroffen werden kann.

Das Carnegie Model ist zwar an sich kein Entscheidungsmodell mit sequenzieller Abfolge, jedoch soll die Kernaussage des Modells, das die Entscheidung auf Organisationsebene potentiell viele menschliche Entscheidungsträger betrifft, im Rahmen des integrierten Entscheidungsmodells dennoch zum Einsatz kommen: Erstens, wenn im Rahmen der Situationsbewertung erkannt wird, dass die Informationslage unzureichend ist und daher eine Entscheidungsunterbrechung nach dem Incremental Decision Model stattfindet. Zweitens, wenn es einer Autorisierung (ebenfalls nach dem Incremental Decision Model) bedarf.

3.2. Beschreibung des Modells

Abbilung 3.1 zeigt das integrierte Entscheidungsmodell, welches durch den Prototypen unterstützt wird, um eine Entscheidungsunterstützung im Rahmen der Abwehr von APT Angriffen zu ermöglichen. Nachdem in der Cyber Security Domain das Observe, Orien, Decide und Act (OODA) Entscheidungsmodell bewährt ist, wurde das erstelle Modell in die bekannten vier Sektionen unterteilt.[CG11, S.2637]

Der Entscheidungsprozess startet damit, dass ein Analysefall erstellt wird. Innherhalb dieses Falles werden durch den Upload von Logfiles alle vorhandenen Informationen gesammelt, um einen Cyberangriff nachweisen zu können. Alle vorhandenen Informationen werden im Rahmen der Informationssuche zusammengetragen. Zum Entwicklungsstand des Prototypen fokussiert sich die Informationssuche auf Netzwerkaktivitäten. Die Informationsbasis kann jedoch alle Bereiche umfassen, die Aufschluss auf ein kompromittiertes System geben können. Als nächster Schritt findet die Auswertung der vorhandenen Informationen statt. Der Prototyp bedient sich hierbei einem individuell erstellen Regelwerk, dass durch das Yara Framework (siehe Abschnitt 5.3.2) realisiert wurde. Wie die Informationsauswertung innerhalb des Prototyps erfolgt, wird in Abschnitt 5.3.3 behandelt. Nach der Informationsauswertung folgt die Situationsbewertung. Sie bildet das Ergebnis der vorhergegangenen Informationssuche und -auswertung, in dem sie Auskunft darüber gibt, ob ein dem Prototyp bekannter APT gefunden wurde, oder nicht.

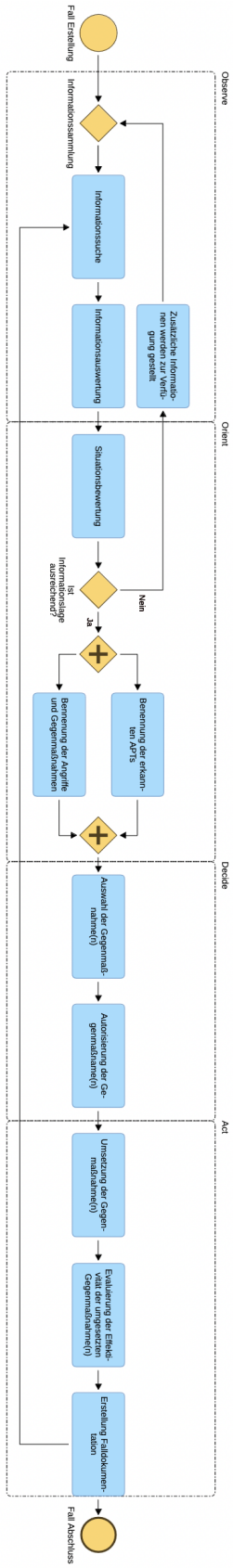


Abbildung 3.1.: Integriertes Entscheidungsmodell

Hier greift das „Incremental Decision Model“ in all seinen drei Varianten. Variante 1 „Simple Match - (if... then)“ findet statt, wenn ein APT erkannt wurde, sprich die Situationsbewertung eine bereits bekannte Situation identifizierte. Sollte kein APT erkannt werden, kommt die Variante zwei zum Einsatz: „Diagnose the Situation - (if (???)... then)“ greift, wenn Informationen in den Logfiles vorhanden sind, die neuartig sind (z.B. weil sie von einer Quelle stammen, zu denen es bis dato keine Yara Regeln gibt), sprich mit denen der Prototyp nicht umgehen kann. Hier bedarf es zusätzlicher Erkennungsregeln, weil die vorhandenen Logfiles nicht ausreichend analysiert werden können. Die dritte Variante „Evaluate Course of Action - (if... then(???))“ kommt ebenfalls zum tragen, wenn ein APT erkannt wurde. Allerdings stellt sich bei genauer Betrachtung der Analyseergebnisse heraus, dass obwohl ein APT erkannt wurde, die getroffenen Gegenmaßnahmen nicht ausreichend geholfen haben. Dies könnte daran liegen, dass der APT fälschlich erkannt wurde (false-positive Ergebnis), oder andere Gegenmaßnahmen notwendig sind, um den Angriff abzuwehren.

Sofern kein APT erkannt wurde, kommt zum ersten mal das „Carnegie Model“ zum Einsatz bzw. eine nicht-programmierte Entscheidung bahnt sich an. Denn um eine möglichst sichere Entscheidung treffen zu können, sollten nun zusätzliche Informationen zur Verfügung gestellt werden. Diese Informationen könnten einerseits weitere Logfiles sein, auf deren Basis die Informationsauswertung erneut starten kann. Andererseits könnten die menschlichen Entscheidungsträger zusätzliche Sensoren, im Sinne von zusätzlichen Logfilequellen zur Verfügung stellen, auf deren Basis dann andere Logfiles bzw. darauf abgestimmte Yara Regeln erstellt werden, um den Suchradius zu vergrößeren und anderen Systemanomalien auf die Spur zu kommen. Generell könnten auch zusätzliche Regeln erstellt werden, um eine größere Anzahl von APTs zu identifizieren. Auf Basis des vorgestellten „Bounded Rationality Model“ muss hier jedoch festgehalten werden, dass die später zu treffende Entscheidung, wie auf eine erkannte Attacke gegengesteuert wird, zu einem gewissen Grad unsicher bleibt, da eine 100% Sicherheit unmöglich zu erreichen ist. Denn selbst wenn alle theoretisch vorhandenen Informationen in die Situationsbewertung einfließen, liegt es in der Natur von APTs, stetig an neuen Angriffen zu arbeiten und dass das vorhandene Sensorium bzw. Regelwerk zunächst nicht zeitgerecht up to date sein kann. Dennoch unterstützt das Entscheidungsmodell bzw. der zugrundeliegende Prototyp dabei, Unsicherheit zu reduzieren, wie im Rahmen des Abschnitt 8 festgehalten wird.

Nachdem zusätzliche Informationen zur Verfügung gestellt wurden, kann eine weitere Iteration der Situationsbewertung stattfinden. Sofern schließlich ein APT vom Prototyp erkannt wurde, benennt er gleichwohl die erkannten Angriffe und die Gegenmaßnahmen, welche sich im Umgang mit der Attacke lt. MITRE ATT&CK bewährt haben.

Nun werden die vorgeschlagenen Gegenmaßnahmen für die gefundenen Angriffe ausgewählt, sprich eine Alternativenselektion findet statt. Wie eine Bewertung der Alternativen, sprich eine Entscheidung über die Auswahl der Alternativen, getroffen wird, ist jedoch nicht Gegenstand des integrierten Entscheidungsmodells. DSS können durch den Einsatz mathematischer Verfahren eine große Unterstützung in der konkreten Entscheidungsfindung bieten. Die Berücksichtigung eines solchen Verfahrens würde jedoch den Rahmen dieser Masterarbeit übersteigen. Dies führt allerdings zu keiner theoretische Einschränkung des Prototypen, da es auch eine bewährte und vergleichsweise simple Methode zur Entscheidungsfindung gibt: die Vor- und Nachteile-Analyse. Sie ist eine qualitative Vergleichsmethode, bei der positive und negative Aspekte jeder Alternative bewertet und verglichen werden. Die Umsetzung ist einfach, da keine mathematischen Kenntnisse erforderlich sind.[RGSM18, S.27] Sofern man sich für ein mathematisches Verfahren entscheiden möchte, sollte jedoch erwähnt werden, das dies kein zu unterschätzendes Unterfangen darstellt. Denn um valide Ergebnisse auf Basis einer Berechenbarkeit bieten zu können, liefert Strong et al. einige Betrachtungskriterien, um Datenkonsumenten eine ausreichende Datenqualität ermöglichen zu können.[SLW02] So fehlt oftmals die Basis für eine Berechenbarkeit, weil die Datenqualität nicht vorhanden ist oder weil die nötigen Daten teilweise nicht vorhanden sind.[SLW02, S.104] Darüber hinaus kann auch die Datenschutzgrundverordnung (DSGVO) potentiell in der Praxis in die Quere kommen, wenn es um die Wahrung der Privatsphäre geht. Ibo van de Poel geht hier näher auf den (vermeintlichen) Konflikt von Sicherheit und Privatsphäre ein, hält jedoch fest, dass es vor allem auf den Kontext ankommt.[CGL20, S.68] Dennoch lässt sich der Konflikt nicht leugnen und erschwert potentiell die Verarbeitbarkeit von Daten. Nach Bevis basieren die Kernfunktion und Technologie innerhalb eines SOC auf Ereignissen von Hunderten oder sogar Tausenden verschiedener Systeme.[Bev13, S.9] Die hierbei produzierten Daten würden sich zwar für Maschine Learning Verfahren anbieten, doch die Komplexität ist außerordentlich hoch und würde den Rahmen der vorliegenden Masterarbeit bei Weitem übersteigen.

Nachdem gewisse Gegenmaßnahmen möglicherweise weitreichende Implikationen innerhalb der betroffenen Organisation haben, bedarf es ggf. einer Autorisierung. Hier findet das „Carnegie Model“ ein zweites mal Berücksichtigung, da bei einer Autorisierung potentiell mehrere Entscheidungsträger zum Einsatz kommen. Im Anschluss werden die genehmigten Gegenmaßnahmen umgesetzt, sprich den Handlungsempfehlungen des Prototypen Folge geleistet. Nach dem Einsatz der Gegenmaßnahmen werden sie anhand ihrer Effektivität evaluiert. Hierzu bietet der Prototyp eine Feedback-Funktionalität an, um die jeweilige Gegenmaßnahme zu bewerten. Schließlich kann eine Dokumentation des Falles heruntergeladen werden, welche eine Aufzeichnung der Fallanalyse beinhaltet (siehe Abschnitt 5.4). Diese Dokumentation dient in weiterer Folge dem organisationellen Lernen betreffend der Informationssuche in einer späteren Prozessinstanz. Denn die Bewertung der Gegenmaßnahmen lässt Rückschlüsse zu, inwieweit sich eine Orga-

nisation ggf. anpassen muss, um auf zukünftige Attacken angemessen vorbereitet zu sein, sofern sie die vorgeschlagenen Gegenmaßnahmen nicht vollumfänglich umsetzen konnten. Der Entscheidungsprozess terminiert schließlich mit dem Abschluss des Falles, was aufgrund von den folgenden vier Szenarien stattfinden kann:

1. Wenn eine zufriedenstellende Lösung gefunden wurde,
2. wenn alle Alternativen ausgeschöpft wurden,
3. wenn alle Erweiterungsmöglichkeiten ausgeschöpft wurden, oder
4. wenn keine zusätzlichen Daten für die Analyse mehr zur Verfügung stehen.

4. Use Cases und Requirements des Prototyps

In diesem Kapitel wird auf die Use Cases und daraus gefolgerten Requirements eingegangen, die dem Design und der Implementierung des Prototypen zugrunde liegen. Zunächst werden die zentralen Use Cases dargestellt (in Diagrammform und tabellarisch). Anschließend werden die Requirements erörtert, die teils aus den Use Cases hergeleitet sind, sortiert in funktionale und nicht-funktionale Requirements.

4.1. Use Cases

Abbildung 4.1 zeigt die Use Cases, die für den Prototyp identifiziert wurden, und ihre Verbindung zueinander. Die Use Cases werden in den Tabellen 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 dokumentiert. Die Dokumentation der Use Cases orientiert sich an dem üblichen Aufbau, wie beschrieben z.B. in [Pro]. Das in einer Use Case Tabelle üblicherweise verwendete Feld „Akteur“ wird weggelassen, da es nur einen relevanten Akteur gibt (den User).

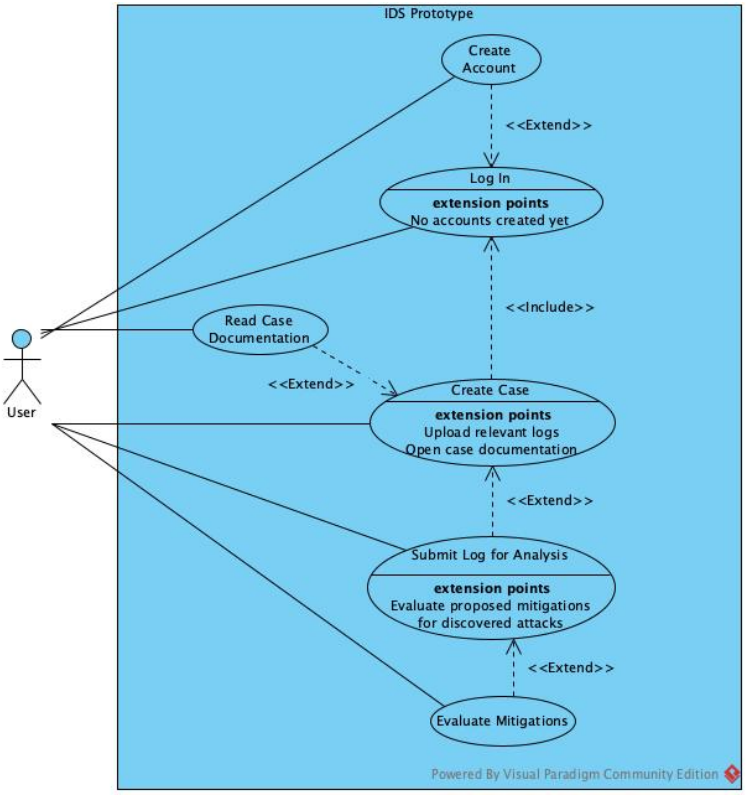


Abbildung 4.1.: Use-Case-Diagramm des Client

ID	U1
Name	Create Account
Beschreibung	Erstellen eines Nutzerkontos, das für den Login verwendet werden kann. Der Login ist Grundvoraussetzung für alle Funktionen des Clients. Das schließt das Erstellen eines Analysefalls ein, der immer einem Nutzerkonto zugeordnet ist.
Verwendete Use Cases	keine
Vorbedingungen	keine
Nachbedingungen	Ein Nutzerkonto wurde erfolgreich erstellt und steht dem Ersteller für den Login bereit.
Standardablauf	Der Nutzer klickt in der Navigation des Clients auf den Link zum Registrierungsformular. Er wird aufgefordert Benutzername und Passwort einzugeben. Nach Absenden des Formulars erscheint eine Erfolgsmeldung und der Nutzer kann sich mit den soeben eingegebenen Zugangsdaten einloggen.
Alternative Abläufe	keine
Fehler	Unerlaubte Zeichen im Username oder Verstöße gegen Passwortvorschriften können zum Scheitern des Registrierungsprozesses führen. Auch der Versuch, als eingeloggter User ein neues Nutzerkonto zu registrieren, wird scheitern (ein eingeloggter User muss sich zuerst ausloggen).

Tabelle 4.1.: Use Case U1: Create Account

ID	U2
Name	Log In
Beschreibung	Erstellen eines Nutzerkontos, das für den Login verwendet werden kann. Der Login ist Grundvoraussetzung für alle Funktionen des Clients. Das schließt das Erstellen eines Analysefalls ein, der immer einem Nutzerkonto zugeordnet ist.
Verwendete Use Cases	keine
Vorbedingungen	Nutzerkonto wurde bereits angelegt, d.h., Verhalten aus U1 erfolgreich durchgeführt.
Nachbedingungen	Nutzer ist eingeloggt und hat Zugriff auf die ihm zugeordneten Daten und alle Funktionalitäten des Clients.
Standardablauf	Nutzer klickt in der Navigationsleiste des Clients auf den Link zum Loginformular, schickt Username und Passwort ab und ist anschließend erfolgreich eingeloggt.
Alternative Abläufe	keine
Fehler	Username existiert nicht, oder das Passwort für einen existierenden User ist falsch. Dann wird der User mit einer Fehlermeldung erneut auf die Login-Seite geleitet.

Tabelle 4.2.: Use Case U2: Log In

ID	U3
Name	Create Case
Beschreibung	Erstellen eines Falls. Ein Fall entspricht einem Verdachtsfall für einen bestimmten Angriff und der Evaluierung dieses Verdachtsfalls durch den Client.
Verwendete Use Cases	U2
Vorbedingungen	Nutzer ist eingeloggt, d.h., Verhalten aus U2 erfolgreich durchgeführt.
Nachbedingungen	Ein neuer, dem User zugeordneter Fall wurde erstellt. Der Fall erscheint als Teil einer Liste von dem User zugeordneten Fällen. Er ist zunächst leer, aber der User kann Logfiles für den Fall hochladen (siehe Tabelle 4.5) und den Fall-Dokumentationslog einsehen (siehe Tabelle 4.4).
Standardablauf	Nutzer klickt in der Navigationsleiste des Clients auf den Link zum Fallerstellungsformular und sendet einen Fallnamen ab. Damit ist der Fall in der Navigationsleiste unter dem Link zur Fallliste einsehbar.
Alternative Abläufe	keine
Fehler	Nutzer klickt in der Navigationsleiste des Clients auf den Link zum Fallerstellungsformular und sendet einen nicht erlaubten Namen ab. Dann wird der User mit einer Fehlermeldung erneut zum Fallerstellungsformular geleitet.

Tabelle 4.3.: Use Case U3: Create Case

ID	U4
Name	Read Case Documentation
Beschreibung	Beziehen einer standardisierten, umfassenden Falldokumentation als Datei.
Verwendete Use Cases	keine
Vorbedingungen	Zumindest ein dem eingeloggten User zugeordneter Fall existiert, d.h., Verhalten aus U3 mindestens einmal erfolgreich durchgeführt.
Nachbedingungen	Aktuelle Falldokumentation als Datei wurde generiert und zum Download bereitgestellt.
Standardablauf	Nutzer klickt in der Navigationsleiste des Clients auf den Link zur Fallliste. Zu dem jeweiligen Fall, für den die Dokumentation gewünscht ist, gibt es einen entsprechenden Downloadlink für die Datei, den der Nutzer anklickt und die aktuelle Dokumentation erhält.
Alternative Abläufe	keine
Fehler	keine

Tabelle 4.4.: Use Case U4: Read Case Documentation

ID	U5
Name	Submit Log for Analysis
Beschreibung	Hochladen einer Logdatei eines möglichen Angriffs zu einem bestimmten Fall, die vom Client automatisch ausgewertet wird.
Verwendete Use Cases	keine
Vorbedingungen	Zumindest ein dem eingeloggten User zugeordneter Fall existiert, d.h., Verhalten aus U3 mindestens einmal erfolgreich durchgeführt.
Nachbedingungen	Logdatei ist auf dem Server. Ihr Inhalt wurde ausgewertet und fließt in die Gesamtauswertung des Falls ein, für den die Logdatei hochgeladen wurde.
Standardablauf	Nutzer klickt in der Navigationsleiste des Clients auf den Link zum Uploadformular. Er wählt den Fall aus, für den er die Logdatei hochladen will, und schickt die Logdatei, die hochgeladen werden soll, im Formular ab. Der Nutzer kann nun die Auswertung in der Übersicht des Falls einsehen, die in der Fallliste verlinkt ist, und in der standardisierten Dokumentation.
Alternative Abläufe	keine
Fehler	Nutzer klickt in der Navigationsleiste des Clients auf den Link zum Uploadformular. Falls noch kein Fall erstellt wurde, kann keiner ausgewählt werden, und der Nutzer erhält eine Fehlermeldung bei Absenden des Formulars. Eine Fehlermeldung wird auch ausgelöst, wenn eine Logdatei hochgeladen wird, die zu groß ist, ein unerlaubtes Format oder einen unerlaubten Namen hat.

Tabelle 4.5.: Use Case U5: Submit Log for Analysis

ID	U6
Name	Evaluate Mitigations
Beschreibung	Einsehen und Bewerten vorgeschlagener Gegenmaßnahmen zu erkannten Attacken für einen Fall.
Verwendete Use Cases	keine
Vorbedingungen	Zumindest ein dem eingeloggten User zugeordneter Fall existiert, für den zumindest eine Logdatei hochgeladen wurde, d.h., Verhalten aus U5 wurde erfolgreich durchgeführt.
Nachbedingungen	Optional wurden die Gegenmaßnahmen vom Nutzer bewertet, ansonsten im Resultat kein Unterschied zu U5.
Standardablauf	Nutzer klickt in der Navigationsleiste des Clients auf den Link zur Fallliste. Er wählt den Fall aus, für den er die Gegenmaßnahmen zu den gefundenen Angriffstechniken einsehen will. Er bewertet eine der auf der Seite des Falls angezeigten Gegenmaßnahmen durch Absenden eines Feedbackformulars.
Alternative Abläufe	Wie Standardablauf, aber ohne Bewertung von Gegenmaßnahmen.
Fehler	keine

Tabelle 4.6.: Use Case U6: Evaluate Mitigations

4.2. Requirements

In den folgenden Abschnitten werden die Requirements des Prototypen aufgezählt.

4.2.1. Funktionale Requirements

Es gibt acht funktionale Requirements, die sich aus den Use Cases ergeben:

- **Requirement 1:** User muss Logfiles hochladen können. Dies soll am Beispiel von Suricata-Logfiles gezeigt werden.
  Relevante Use Cases: U5

- **Requirement 2:** Anhand von Logfiles soll bei einem Angriff der Angreifer identifiziert werden (zwischen APT19, APT28 und TURLA). Dies soll am Beispiel von Suricata-Logfiles gezeigt werden.
  Relevante Use Cases: U3, U5

- **Requirement 3:** Anhand von Logfiles sollen Angriffstechniken gemäß MITRE ATT&CK identifiziert werden. Dies soll am Beispiel von Suricata Logfiles gezeigt werden.
  Relevante Use Cases: U5

- **Requirement 4:** Der Client soll eine „MITRE ATT&CK"- Datenbank beinhalten zu einem gegebenen Stand und auf diese zugreifen.
  Relevante Use Cases: U5, U6

- **Requirement 5:** Der Client soll, anhand der Identifikation von Techniken und Angreifer, Gegenmaßnahmen aus der „MITRE ATT&CK"-Datenbank vorschlagen.
  Relevante Use Cases: U3, U6

- **Requirement 6:** Die Gegenmaßnahmen sollen vom User einzeln bewertet werden können.
  Relevante Use Cases: U6

- **Requirement 7:** Der Client soll eine Dokumentation generieren, die Informationen über die eingelesenen Files, identifizierte Techniken und Angreifer, empfohlene Gegenmaßnahmen bzw. deren Bewertung vom User enthalten.
  Relevante Use Cases: U4

- **Requirement 8:** Die eingegebenen Daten des Users werden mittels Login gegen unbefugten Zugriff abgesichert.
  Relevante Use Cases: U1, U2

4.2.2. Nichtfunktionale Requirements

Es gibt neun nichtfunktionale Requirements:

- **Requirement 9:** Die Interaktion mit dem Client soll für den User selbsterklärend sein.
- **Requirement 10:** Der Client ist ein Webclient und somit betriebssystemunabhängig ausführbar.
- **Requirement 11:** Der Client wird in Python programmiert.
- **Requirement 12:** Der Client verwendet YARA-Regeln für die Identifikation von Techniken und Angreifern.
- **Requirement 13:** Der Client muss mit sämtlichen Abhängigkeiten in einem Docker-Container enthalten und über die Ausführung eines Docker-Compose-Files installierbar sein.
- **Requirement 14:** Der Client muss modular aufgebaut sein, um Verständlichkeit, Erweiterbarkeit und Test-/Wartbarkeit zu verbessern.
- **Requirement 15:** Der Client soll mit einer non-relationalen (z.B. MongoDB) Datenbank (für Techniken, Angreifer, Gegenmaßnahmen gemäß MITRE ATT&CK) im Backend laufen.
- **Requirement 16:** Der Zugriff auf den Client muss über einen Webserver (z.B. NGINX) erfolgen.
- **Requirement 17:** Der Client muss abgesichert sein gegen Injection-Angriffe.

5. Design des Client-Prototyps

In diesem Kapitel wird auf das Design des Client-Prototyps eingegangen, das auf Grundlage der Requirements, siehe 4.2, entworfen wurde.

5.1. Geplante Nutzung des Clients

Die Grundfunktionalität des Client-Prototyps ist es, einem Nutzer zu ermöglichen, Logdateien von vermuteten Angriffen hochzuladen und zu analysieren. Der Client gleicht den Inhalt dieser Logdateien mit ihm bekannten Angriffsmustern einer bestimmten Menge möglicher Angreifer (APTs) ab. Werden Angriffsmuster gefunden, bietet der Client eine Einschätzung, um welchen APT es sich handeln könnte. Zusätzlich werden zu den identifizierten Angriffstechniken technische Maßnahmen vorgeschlagen, die der User ergreifen kann, um der Bedrohung durch die konkreten Techniken entgegenzutreten. Die Wirksamkeit dieser Maßnahmen kann vom User im Einzelnen beurteilt werden. Der User kann für eine detailliertere Analyse eines Cases, einschließlich hochgeladener Logdateien, ihrer Analyse, vorgeschlagener Gegenmaßnahmen und ihrer Bewertungen, eine stets aktuelle standardisierte Dokumentationsdatei herunterladen, die den gesamten Case abbildet.

5.2. Grundstruktur des Clients

Alle Funktionen, die der User ausführen kann, sind durch ein intuitiv verständliches grafisches User-Interface verwendbar. Der Client-Prototyp ist als Web-Applikation konzipiert und somit plattformunabhängig. Diese Web-Applikation soll in einem Container bereitgestellt werden, ebenfalls aus Gründen der Plattformunabhängigkeit. Der Client ist vom User über einen Webserver erreichbar, der auch mit höherer Zugriffslast umgehen kann. Die Daten, die vom Client verwendet oder vom User bereitgestellt werden, werden in einer nichtrelationalen Datenbank gespeichert, deren Struktur über einen Object-Document-Mapper mit dem Domain Model der Applikation abgestimmt wird (siehe Abschnitt 5.2.2). Ein Teil der Daten liegt bei Aufruf des Clients bereits vor. Das ist die „MITRE ATT&CK Enterprise“-Datenbank in der Version 9.0.[Theh] Sie enthält Informationen über APTs und Malware-Programme, die von ihnen verwendeten Angriffstechniken sowie mögliche Gegenmaßnahmen. Beruhend auf diesen Informationen wird dem User eine Analyse eines möglichen Angriffsfalles bereitgestellt.

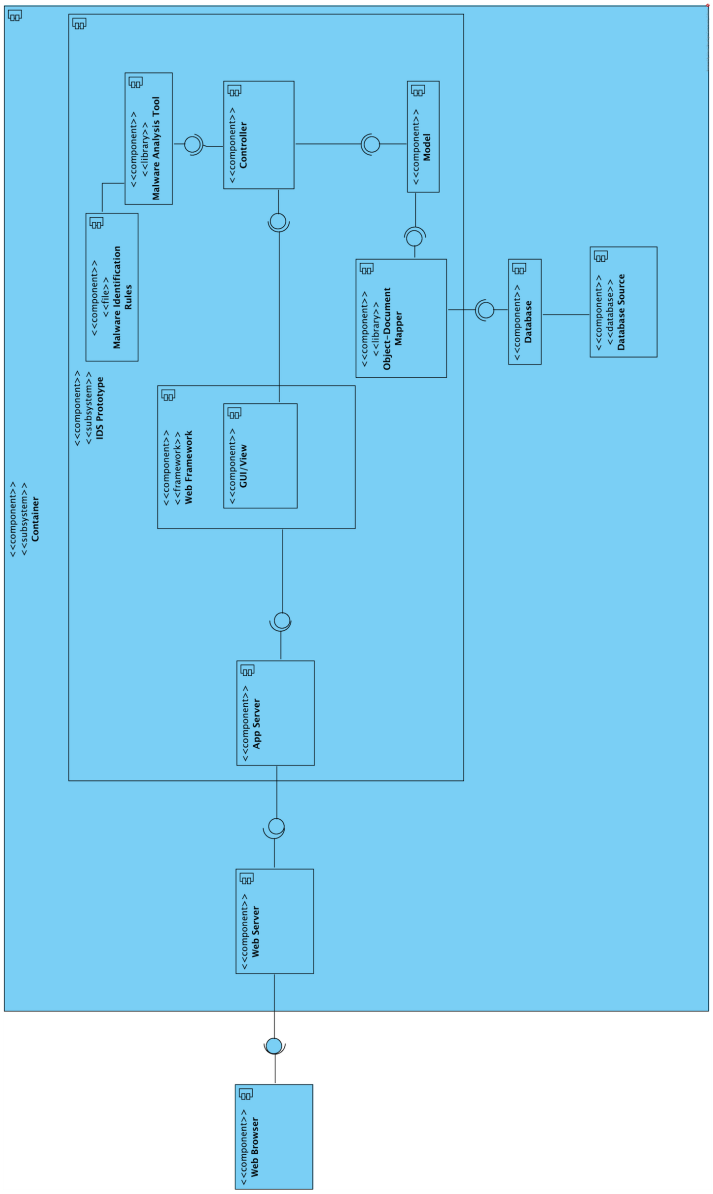


Abbildung 5.1.: Komponentendiagramm des Clients

Die MITRE-Datenbank bietet keine direkten Regeln, um mögliche Angriffe zu identifizieren. Sie bietet jedoch die Hintergrundinformationen, die es braucht, um derartige Regeln zu entwerfen. Diese Regeln werden im Client definiert und bereitgestellt für die Analyse hochgeladener Logfiles. Im Komponentendiagram in Abb. 5.1 wird die genauere Struktur des Clients dargestellt und in Abschnitt 5.2.1 beschrieben.

Der Client wird nach den Prinzipien der objektorientierten Programmierung entworfen. Er beruht auf einem Model-View-Controller-Pattern. Dem Model entsprechen die grundlegenden Klassen bzw. Datentypen, die den Zustand des Client-Prototyps abbilden. Der View entsprechen die Funktionalitäten (im Client definiert als Methoden), die der Nutzer des Prototyps über die Web-Oberfläche aufrufen kann. Über die Verwendung der Controller-Klassen können durch die View-Methoden Objekte des Model bearbeitet werden oder der aktuelle Zustand des Model abgerufen werden. Teilweise werden auch in Controller-Klassen andere Controller-Klassen direkt aufgerufen, z.B. wenn gemäß dem Model Abhängigkeiten bestehen und die Veränderung eines Model-Objektes die Veränderung eines anderen nach sich zieht. Das heißt, Controller können nicht nur von View-Methoden genutzt werden, sondern auch voneinander. Die Controller-Klassen sind eng mit einzelnen Model-Klassen verbunden. Beispielsweise werden Objekte der Klasse `APT` über Methoden einer Instanz des `APTController` bearbeitet.

5.2.1. Beschreibung der System-Komponenten

Im Folgenden wird die Funktion der einzelnen System-Komponenten gemäß Diagramm genauer beschrieben:

- **Web Browser**: Über diesen findet userseitig der Zugriff auf die Komponente **Container** statt, konkret über die Subkomponente **Web Server**, auf die der Web Browser zugreift. Sie ist der Einstieg in die Nutzung des Clients (über die Verbindung von Web Server zu **App Server**, der eine Subkomponente der Komponente **IDS Prototype** ist – dem Client, siehe unten).

- **Container**: Der Container bündelt die Anwendung mit all ihren Bausteinen, d.h., sowohl die Anwendungslogik als auch die Deploymentfunktionalitäten und andere zum Funktionieren der Anwendung erforderliche Elemente, in einem ausführbaren Paket.[Doce] Der Container bildet ein System aus folgenden Subkomponenten:
  - **Web Server**: Dieser vermittelt zwischen der **Web Browser**-Komponente, d.h. dem Außenzugriff und der Anwendung, auf die zugegriffen werden soll. Der Web Server empfängt Anfragen seitens Externer (den Usern) mittels Web Browser und sendet Antworten seitens des Clients. Er übernimmt auch wichtige Funktionen für eine Webanwendung, z.B. Nebenläufigkeit von Zugriffen oder Lastverteilung zwischen Zugriffen.

Er kommuniziert mit der Subsystem-Komponente **IDS Prototype** über dessen Subkomponente **App Server** (siehe unten).

- **IDS Prototype**: Hierbei handelt es sich um den konkreten Client, d.h., die Komponente, die sämtliche für die Anwendung spezifische Business-, Datenbankzugriffs- sowie Darstellungslogiken bereitstellt. Dieses Subsystem umfasst folgende Subkomponenten:
  - **App Server**: Über diese Komponente kommuniziert der **Web Server** mit dem Client und umgekehrt. Ein App Server ist ein wesentlicher Bestandteil von Python-Webanwendungen. Er standardisiert die Kommunikation zwischen den Webanwendungen und Webservern.[Ken] Es ist übliche Praxis, vor den App Server einen Web Server zu setzen, über den auf den App Server und somit auf die Anwendung zugegriffen werden kann. Grund hierfür ist, dass Web Server im allgemeinen robuster und bei hoher Last performanter sind.[Digb] Der App Server sendet Anfragen sowie empfängt Antworten vom **Web Framework**.
  - **Web Framework**: Diese Framework-Komponente beschreibt das Framework, welches verwendet wird, um die Webanwendung zu erstellen. Sie stellt die Anwendungslogik dem **App Server** bereit und organisiert die restlichen Komponenten des Clients je nach Art des externen Inputs. Das Web Framework enthält eine Subkomponente:
    - **GUI/View**: Diese Komponente ermöglicht den Zugriff auf die vom Web Framework bereitgestellten Funktionalitäten über eine grafische Oberfläche. Sie wird implementiert mit Mitteln, die das Web Framework selbst bereitstellt, und ist daher eine Subkomponente des Web Framework.
  - **Controller**: Diese Komponente stellt Methoden bereit, um Datenbankinhalte zu schreiben oder zu lesen, und implementiert die Anwendungslogik.
  - **Malware Analysis Tool**: Diese Library-Komponente bildet die Malwareanalysefunktionalitäten ab, die von den Controllern verwendet werden.
  - **Malware Identification Rules**: Auf diese Datei-Komponente wird vom **Malware Analysis Tool** bei der Analyse zugegriffen. Es handelt sich dabei um eine Sammlung von Erkennungsregeln gemäß der vom Tool geforderten Syntax in Form einer Textdatei.
  - **Model**: Diese Komponente bildet das Domain Model ab, auf das die **Controller**-Komponente schreibend und lesend zugreifen kann.

    - **Object-Document Mapper**: Diese Komponente ist die Schnittstelle zwischen der **Database**-Komponente einerseits und der **Model**-Komponente andererseits. Sie regelt den Austausch zwischen der Datenbank und der Anwendung, indem sie Objekte aus der Datenbank auf Objekte aus dem Anwendungsmodel abbildet und umgekehrt.
  - **Database**: Diese Komponente steht für die Implementierung der Anwendungsdatenbank. Sie stellt die Inhalte der **Database Source**-Komponente über die **Object-Document Mapper**-Komponente dem Client, d.h. der **IDS Prototype**-Komponente, zur Verfügung.
  - **Database Source**: Diese Datei-Komponente enthält die Inhalte der Anwendungsdatenbank.

5.2.2. Domain Model des Clients

Das Klassendiagramm in Abb. 5.2 stellt das Domain Model des Clients dar. Die einzelnen Klassen aus dem abgebildeten Modul „ids_prototype.classes“ werden in der folgenden Auflistung genauer beschrieben (das Modul `flask_login` wird in Kapitel 6 beschrieben, da seine Verwendung sich aus der konkreten Implementierung ergibt):

- **UserAccount:** Der UserAccount schützt vom User eingegebene Daten vor unbefugtem Zugriff. Ein UserAccount kann mehrere Cases erstellen, die Verdachtsfälle für einen Angriff abbilden. Zu diesen Cases kann er Logfiles hochladen. Er kann eine Dokumentation für einen Case beziehen in Form einer Datei. Zu vorgeschlagenen Gegenmaßnahmen bzgl. einer für einen Case identifizierten Attacke kann er Feedback abgeben.

- **Case:** Dabei handelt es sich um einen bestimmten Verdachtsfall für einen Angriff. Der Case ist eine Sammlung von Logfiles, von welchen der User den Verdacht hat, dass sie einen gemeinsamen Angriff beschreiben. Der User kann beliebig viele Cases erstellen. Für jeden Case kann der User beliebig viele Logfiles hochladen. Für einen Case gibt es eine aggregierte Fallanalyse, die sich zusammensetzt aus den Einzelanalysen der Logfiles. Aus der Analyse geht hervor, für wie wahrscheinlich der Client es erachtet, dass ein bestimmter APT aus einer gegebenen Menge von APTs der Angreifer ist (bzw. ob überhaupt verdächtige Aktivität festgestellt wird). Mit der Analyse gehen Vorschläge zu Gegenmaßnahmen (Mitigations) einher, die sich auf bestimmte Techniken (Techniques) beziehen. Die Mitigations können vom User auf ihre Nützlichkeit hin bewertet werden (Feedback). Zu jedem Case kann eine aktuelle Dokumentation als Datei heruntergeladen werden mit standardisierten Feldern (der Inhalt der Dokumentation wird im Abschnitt 5.4 genauer beschrieben).

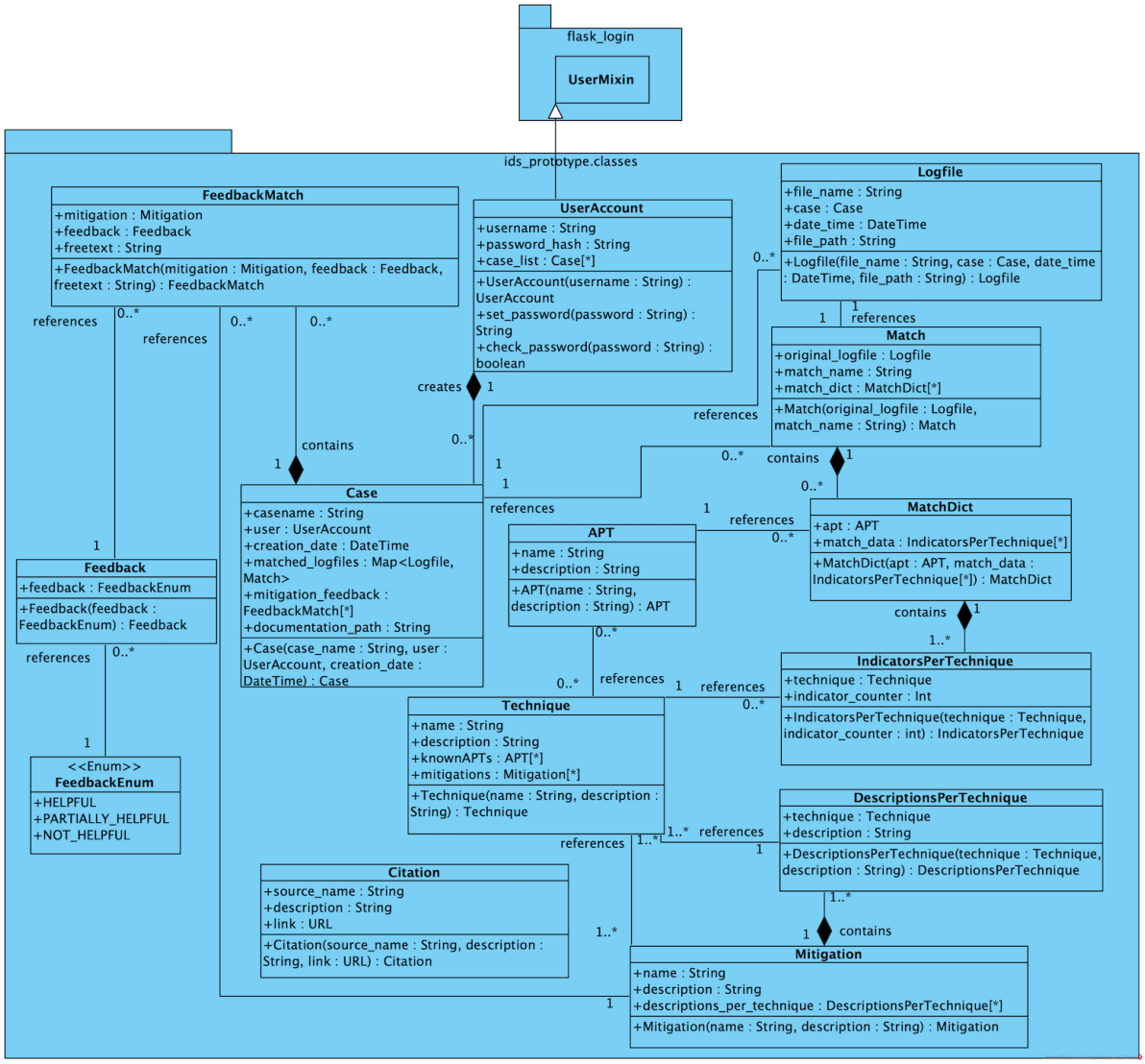


Abbildung 5.2.: Klassendiagramm für das Domain Model des Clients

- **Logfile:** Dabei handelt es sich um die Repräsentation hochgeladener Logdateien in der Datenbank. Sie können zu einem bestimmten Case hochgeladen werden. Beim Upload werden sie automatisch auf Angriffsindikatoren analysiert. Die zu einem Logfile gehörige Analyse ist abgebildet in einem Objekt der Klasse Match (die Logfiles werden abgeglichen, d.h. „matched“ mit einem Satz an Regeln zur Erkennung von Angriffsindikatoren).
- **Match:** Ein Match entspricht einem hochgeladenen Logfile. Es enthält alle Attacken, die in einem Logfile identifiziert worden sind, in Form eines MatchDict.
- **MatchDict:** Im MatchDict wird für jeden erkannten APT eine Liste von IndicatorsPerTechnique-Instanzen gespeichert.
- **IndicatorsPerTechnique:** Eine IndicatorsPerTechnique-Instanz bildet ab, wie viele Indikatoren (d.h. einzelne Logzeilen, die mit einer Erkennungsregel für einen Angriffsindikator übereinstimmen) für eine bestimmte Technik gefunden wurden. Die Indikatoren in der IndicatorsPerTechnique-Instanz entsprechen nicht nur der Technik, sondern auch dem APT, dem die IndicatorsPerTechnique-Instanz im MatchDict zugeordnet ist.
- **APT:** Ein APT steht für einen *Advanced Persistent Threat* gemäß der „MITRE ATT&CK Enterprise“-Datenbank. Es handelt sich dabei um einen Angreifer. Der Angreifer kann bei seinem Angriff verschiedene Techniken anwenden. Ziel des Clients ist, durch Analyse der Logfiles zu bestimmen, welcher Angreifer hinter identifizierten Techniken und hinter dem Angriff insgesamt stecken könnte.
- **Technique:** Eine Technique steht für eine Angriffstechnik, die durch einen APT angewendet wird. Die Techniques werden von der „MITRE ATT&CK Enterprise“-Datenbank definiert und bezogen. Zu einer Technique gibt es Gegenmaßnahmen in Form von Mitigation-Instanzen.
- **Mitigation:** Eine Mitigation steht für eine Gegenmaßnahme zu einer bestimmten Technique. Mitigations werden von der „MITRE ATT&CK Enterprise“-Datenbank definiert und bezogen. Sie haben eine allgemeine Beschreibung und eine spezifische Beschreibung je Technique, die aus konkreten Handlungsvorschlägen besteht. Die spezifischen Beschreibungen je Technique werden als Liste von DescriptionsPerTechnique-Instanzen dargestellt.
- **DescriptionsPerTechnique:** Jeder Technique wird ihre spezifische Beschreibung für die Mitigation zugeordnet, der das DescriptionsPerTechnique-Objekt zugeordnet ist.
- **FeedbackMatch:** Eine FeedbackMatch-Instanz stellt das User-Feedback für eine Mitigation dar, die in einem bestimmten Case vorgeschlagen wird.

Eine Mitigation kann über eine festgelegte Bewertungsskala und über Freitext bewertet werden.

- **Feedback:** Bildet einen möglichen Eintrag der festgelegten Bewertungsskala für Mitigations ab.
- **FeedbackEnum:** Definiert alle möglichen Einträge der Bewertungsskala für Mitigations.
- **Citation:** Instanzen dieser Klasse bilden Verweise aus den Beschreibungstexten für Technique- und Mitigation-Instanzen ab. Die Citation-Instanzen werden von der „MITRE ATT&CK Enterprise“-Datenbank bezogen. Sie ordnen einer Referenz eine URL zu. Auch die Beschreibungstexte für Technique- und Mitigation-Instanzen werden von der „MITRE ATT&CK Enterprise“-Datenbank bezogen. Für die Darstellung im Client werden die Referenzen in den Beschreibungstexten mit den zu den Referenzen gehörigen Links hinterlegt.

5.2.3. Controller-Klassen

Wie einleitend beschrieben, findet die Veränderung des Models über Controller-Klassen statt. Die Verbindung der Controller-Klassen zu den Model-Klassen sowie der Controller-Klassen untereinander wird im Diagramm aus Abb. 5.3 abgebildet. Einzelne Controller behandeln keine konkreten Model-Klassen, sondern implementieren technische Funktionalitäten des Clients, die von den View-Methoden aufgerufen werden. Diese sind:

- **RepoController:** Diese Controller-Klasse bietet eine Methode, um die Datenbank des Clients mit den Daten aus der im STIX2[OAS]-Format bereitgestellten „MITRE ATT&CK Enterprise“-Datenbank zu füllen (die Implementierungsdetails und damit auch das ebenfalls im Diagramm abgebildete Modul „stix2“ werden in Abschnitt 6 näher behandelt). Sie bildet die für die Funktionen des Clients nötigen Instanzen aus den MITRE-Daten (z.B. die STIX2-Äquivalente zu APTs und Techniken) ab auf Instanzen der entsprechenden Model-Klassen des Clients. Die Methode wird nur dann aufgerufen, wenn die Datenbank leer ist, d.h. zur Erstinstantiierung des Clients.
- **UploadController:** Diese Controller-Klasse bietet eine Schnittstelle zwischen der View-Methode für den Upload eines Logfile durch den User und der Applikationslogik, die durch den Upload getriggert werden soll (u.a. Speichern des Logfiles, Auswerten des Logfiles).
- **LoginController:** Diese Controller-Klasse stellt die erforderliche Applikationslogik für die View-Methode für den Login des Users bereit. Sie fungiert damit auch als Schnittstelle zwischen der View und den Model-Instanzen der UserAccount-Klasse, in denen z.B. das Passwort hinterlegt ist.

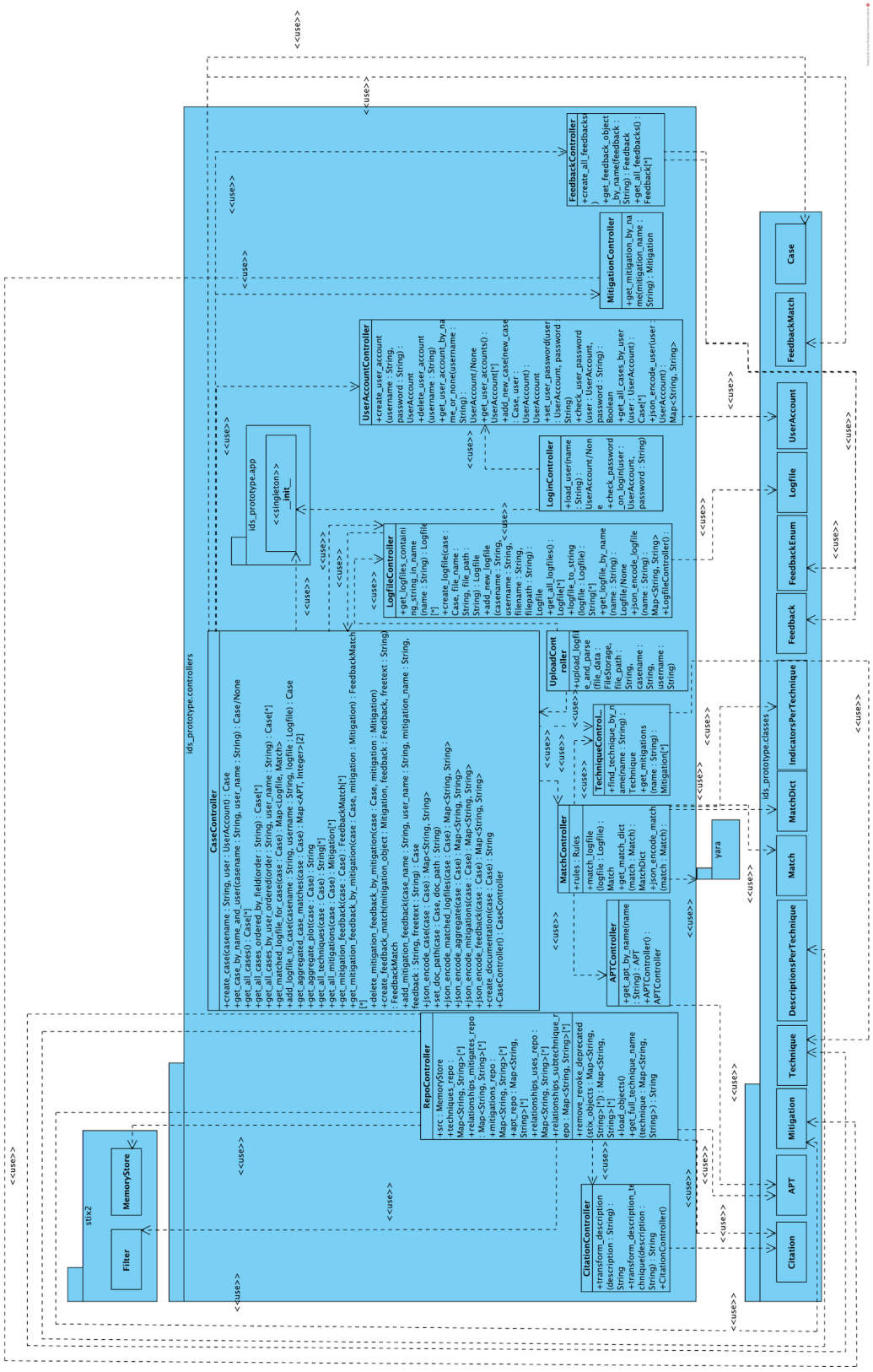


Abbildung 5.3.: Klassendiagramm für die Controller-Klassen des Clients

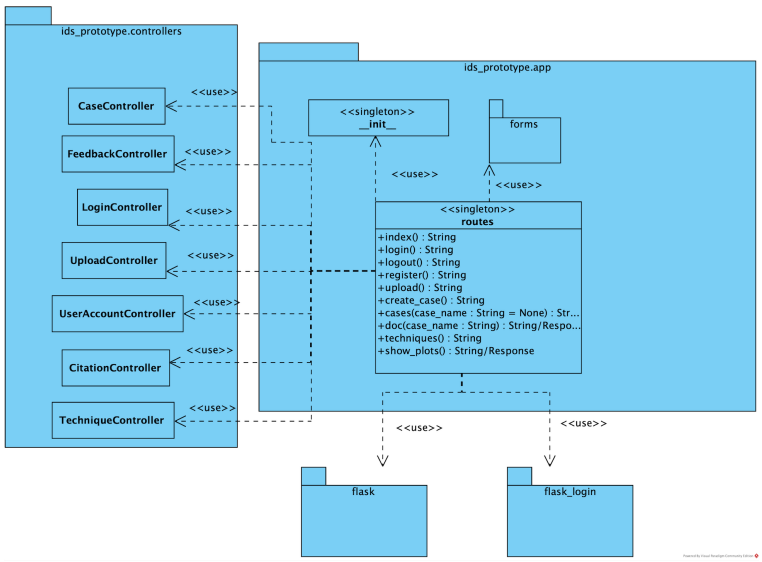


Abbildung 5.4.: Klassendiagramm für die View-Methoden des Clients

5.2.4. View-Methoden

Die Verbindung der View-Methoden (dargestellt in dem Modul `routes`) mit den Controllern (und über die Controller mit den Model-Klassen) wird im Diagramm aus Abb. 5.4 abgebildet. Die View-Methoden greifen nicht auf alle Controller direkt zu. Auf manche Domain-Klassen greifen die View-Methoden indirekt zu, indem von ihnen aufgerufene Controller wiederum die relevanten Controller für diese Domain-Klassen aufrufen. Der `RepoController` wird von den View-Methoden nie aufgerufen, da er nur zur Erstinitialisierung des Clients verwendet wird, bevor überhaupt eine View dargestellt wird.

Die Details der Implementierung, wie das „forms“-Modul und die Flask-bezogenen Module, werden im Abschnitt 6 genauer erläutert.

5.3. Hochladen und Auswerten von Logdateien

Die grundlegende Zielsetzung des Clients ist es, auf einem Windows-PC einen Angriff durch einen APT anhand von Logdateien, die PC-Aktivitäten abbilden, identifizieren zu können. Zusätzlich bietet der Client eine Einschätzung, welcher konkreter APT verantwortlich sein könnte. Schließlich bietet der Client Gegenmaßnahmen an für die gefundenen Angriffstechniken, die vom User bewertet werden können.

Das Hochladen und Auswerten von Logdateien ist die Kernlogik des Clients. Ihr Ablauf wird daher im Folgenden näher beschrieben.

Zum Test des vorliegenden Prototyps wurden drei repräsentative APTs ausgewählt:

- **APT19** [Theb], ein chinesischer APT, das unter anderem über Phishing-Kampagnen diverse strategische Industriesektoren angreift,
- **APT28** [Thec], ein russischer APT, von den unter anderem angenommen wird, dass es in den amerikanischen Präsidentschaftswahlkampf 2016 eingegriffen hat,
- **Turla** [Thei], ein russischer APT, das seit 2004 Angriffe in über 45 Ländern durchgeführt hat und diverse sicherheitskritische Industrien angegriffen hat.

Sobald neue Yara-Erkennungsregeln für weitere APTs zur Verfügung stehen, können diese APTs hinzugefügt werden (der Aufbau dieser Regeln wird in Abschnitt 5.3.2 beschrieben).

5.3.1. Logformate als Input

Ein Angriff durch einen APT kann sich auf zahlreichen Ebenen eines Netzwerks oder Computers ereignen. Entsprechend können sich Indikatoren eines Angriffs auch in vielen verschiedenen Logs niederschlagen. Lokal wären dies z.B. PowerShell-Logs oder die Security-Logs des Betriebssystems. Ein Angriff hinterlässt auch Spuren in der Netzwerkaktivität – z.B. durch das Senden von Anweisungen durch den Angreifer an die installierte Schadsoftware am infizierten PC oder durch das Extrahieren sensibler Daten am infizierten PC durch den Angreifer.

Der Fokus liegt im Prototyp auf der Analyse der Netzwerkaktivität. Diese kann über das Tool *Wireshark*[Wir] aufgenommen werden in Form einer „Packet Capture“ (kurz *PCAP*). PCAP-Dateien bilden sämtliche ein- und ausgehenden Netzwerkpakete vollständig ab. Das Bedrohungserkennungstool *Suricata*[Thej] wertet PCAP-Dateien aus und stellt anhand seines Erkennungsregelsatzes eine Logdatei bereit über alle in der Auswertung erkannten Angriffsindikatoren.

Diese Suricata-Logdatei, im *fast*-Format, ist gegenwärtig das einzig erlaubte bzw. auswertbare Logformat für den Upload. Die Logdatei ist wiederum die Grundlage für die Auswertung des vermuteten Angriffs durch den Client. Der Client ordnet die identifizierten Angriffe bestimmten Techniken und bestimmten APTs zu. Jede Suricata-Logzeile steht für eine Angriffstätigkeit.

5.3.2. YARA-Regeln zur Logauswertung

Die Zuordnung der Angriffstätigkeit durch den Client wird mithilfe von YARA-Regeln vorgenommen. YARA[Yar] ist ein auf die Erkennung von Schadsoftware ausgelegtes Framework, mit dem Regeln nach einem bestimmten Format definiert werden können. Ein Beispiel für eine solche Regel, wie sie im Client vorkommt, wird in Abb. 5.5 dargestellt. Sie besteht aus konkreten Strings, die in einem Logfile gematcht werden sollen, und einer Bedingung für die Aktivierung der Regel. Im

```
rule DealersChoice : APT28
{
    meta:
        technique1="Application Layer Protocol: Web Protocols"

    strings:
        $dealerschoice="DealersChoice" nocase

    condition:
        $dealerschoice

}
```

Abbildung 5.5.: Beispiel für eine YARA-Regel aus dem Client (hier zum Finden von Indikatoren für die Verwendung der Malware „DealersChoice“ durch APT28)

Client ist diese Bedingung im Allgemeinen, dass mindestens eine der String-Variablen gematcht wird. Zudem besteht sie aus optionalen Metadatenfeldern unter dem Abschnitt „meta“, die im Client zur weiteren Analyse genutzt werden können. Für die weitere Analyse durch den Client wird ein Metadateneintrag „technique1“ (optional „technique2“, „technique3“ etc., falls mehrere Techniken durch einen bestimmten Indikator angezeigt werden) definiert. Dieser Eintrag steht für die Angriffstechnik gemäß MITRE, die vom Indikator angezeigt wird, der von der betreffenden YARA-Regel abgedeckt wird. (Für die Erkennung von Angriffsaktivitäten durch die YARA-Regeln in Logfiles sind die Metadatenfelder nicht relevant. Sie werden ausschließlich zur Erstellung der Analyse durch den Client herangezogen, damit dieser die jeweiligen Indikatoren den jeweiligen Techniken, die die Indikatoren anzeigen, zuordnen kann.) Diese Zuordnung zwischen Indikatoren und Angriffstechniken ist das Ergebnis von Recherche über die Angriffsmuster der behandelten APTs, ausgehend von den von MITRE bereitgestellten Informationen und Verweisen.

Ähnlich verhält es sich mit sogenannten „Tags“. Diese folgen dem Namen der YARA-Regel und sind von ihm durch einen Doppelpunkt getrennt. Vom Client-Prototyp werden sie genutzt, um den APT oder die APTs zu nennen, die vom Indikator angezeigt werden, der von der YARA-Regel abgedeckt wird. Auch diese Zuordnung beruht auf den von MITRE bereitgestellten Daten. So ist von MITRE etwa für jede Angriffstechnik verzeichnet, wie konkret die Umsetzung der Technik aussieht bei einem bestimmten APT. Entsprechend unterscheiden sich die Indikatoren, die im Suricata-Log gefunden werden können, je nach APT, selbst bei Anwendung der gleichen Technik. APT19 verwendet z.B. die Technik „Application Layer Protocol: Web Protocols“. Das bedeutet aber nicht, dass jede YARA-Regel, die die Technik „Application Layer Protocol: Web Protocols“ indiziert, auch einen APT19-Angriff indiziert. Die YARA-Regel kann z.B. auch nur die Technik „Application Layer Protocol: Web Protocols“ in ihrer Anwendung

durch APT28 anzeigen. Daher ist ein Treffer für einen Indikator für einen Angriff durch einen bestimmten APT immer durch beide Aspekte der YARA-Regel definiert: Durch die Angriffstechnik im „meta“-Abschnitt und durch das Vorkommen des APT-Namens in den Tags.

5.3.3. Auswertungsergebnisse im Client

Die Auswertung eines Logfile wird direkt durch seinen Upload ausgelöst. Um ein Logfile uploaden zu können, ist es userseitig erforderlich, zunächst einen Account zu registrieren und sich im Anschluss im Client einzuloggen. Der User kann dann einen Fall erstellen. Erst dann kann ein Logfile hochgeladen werden, da ein Logfile einem Fall eines Users zugeordnet werden muss. Es handelt sich bei jedem hochgeladenen Logfile für einen Fall um eine Informationsquelle zu diesem Fall. Die Logfiles gruppiert in einem Fall stehen für verschiedene Aspekte eines bestimmten Angriffs, den der User vermutet und den er mittels des Clients näher analysieren möchte. Für einen weiteren Angriff wäre ein neuer Fall anzulegen.

Jede Analyse eines Logfile mittels der YARA-Regeln liefert eine Datenstruktur, die enthält, welche YARA-Regeln bei der Analyse angeschlagen haben. Diese Datenstruktur wird vom Client ausgewertet, um sie dem User zu präsentieren. Ziel der Analyse ist erstens, für jeden APT herauszufinden, welche Angriffstechniken für ihn gefunden wurden, und zweitens, wie viele Indikatoren für die jeweiligen Angriffstechniken des APT gefunden wurden. Mit „Indikator“ ist im Endergebnis keine YARA-Regel gemeint und auch keiner der Einzelindikatoren (z.B. Strings), die in den YARA-Regeln festgelegt sind als Kriterium für das Feuern einer YARA-Regel. Ein Indikator im Endergebnis entspricht einer Logzeile, in der die Angriffstechnik für den APT identifiziert wurde. Der Grund dafür ist, dass eine Logzeile eine bestimmte Aktivität abbildet.

Die Analyse eines Logfile findet daher Logzeile für Logzeile statt und wird durch die Methode `match_logfile` einer MatchController-Instanz durchgeführt. Jede Logzeile wird auf Übereinstimmung mit den YARA-Regeln geprüft. Die Analyse nimmt zunächst die Form eines Array an. Dieses Array speichert pro Logzeile im Logfile die Match-Datenstruktur, die vom YARA-Framework bei Prüfung auf Übereinstimmung mit den YARA-Regeln retourniert wird. Die Vorgangsweise hat den Zweck, dass pro Logzeile eine Angriffstechnik immer nur höchstens einmal identifiziert wird. Es kann passieren, dass in ein und derselben Logzeile Einzelindikatoren (z.B. Strings) aus verschiedenen YARA-Regeln existieren, die YARA-Regeln aber derselben Technik zugeordnet sind. Dann würde die retournierte YARA-Datenstruktur zwei Übereinstimmungen für eine Technik in einer Logzeile anzeigen, obwohl es sich bei einer Logzeile um eine geschlossene Angriffshandlung handelt.

Im Anschluss an die Auswertung der einzelnen Logzeilen wird die Auswertung pro Logzeile nach Technik sortiert (mittels der oben beschriebenen „meta“-Einträge einer jeden YARA-Regel). Jeder Technik wird in diesem Schritt zugeordnet, welche YARA-Regeln für sie in der gegebenen Logzeile angeschlagen

Abbildung 5.6.: Beispiel für ein Objekt der Klasse „MatchDict“, wie abgebildet in der MongoDB-Instanz (Screenshot aus dem Tool „MongoDB Compass“[Monb])

haben. Das ist ein Zwischenschritt, um im Anschluss mehrere angeschlagene YARA-Regeln für dieselbe Technik in einer einzelnen Logzeile, wie beschrieben, als nur einen Indikator zu zählen. Aus den nach Technik sortierten YARA-Regeln pro Logzeile wird im Anschluss ausgelesen, für welchen APT oder welche APTs sie ein Indikator sind. Dies geschieht mittels der oben beschriebenen „Tags“ einer jeden YARA-Regel.

Auf dieser Grundlage wird die Gesamtauswertung sortiert nach APTs vorgenommen. Jedem APT wird zugeordnet, welche entsprechenden Techniken gefunden wurden, und wie viele Indikatoren (d.h. Logzeilen als abgeschlossene Angriffsaktivität, für die eine oder mehrere YARA-Regeln für die jeweilige Technik angeschlagen haben) für die jeweilige Technik des APT gefunden wurden. Diese Endauswertung hat die Form einer „MatchDict“-Instanz. Die einzelnen Mappings von Techniken zur Anzahl der Logzeilen haben die Form von „IndicatorsPerTechnique“-Instanzen. Sie sind in der „MatchDict“-Instanz nach den APTs sortiert, welche die Technik laut den Ergebnissen der Logfileanalyse angewendet haben.

Die Struktur des Auswertungsergebnisses für ein Logfile in Form einer „MatchDict“-Instanz ist beispielhaft abgebildet in Abb. 5.6. Bei den als „Object“ bezeichneten Instanzen im Array „match_data“ handelt es sich um „IndicatorsPerTechnique“-Instanzen.

5.4. Darstellung der Analyseergebnisse und Feedback

Beruhend auf der in Abschnitt 5.3.3 beschriebenen Auswertung wird dem User eine Darstellung der Analyseergebnisse präsentiert.

5.4.1. Darstellung im Client

Sie findet auf Fallebene statt und es gibt pro Case eine Seite im Client. Die Darstellung findet einerseits aggregiert statt: Der User kann sehen, wie viele Techniken pro APT über alle Logfiles eines gesamten Falls identifiziert wurden. Teil dieses Überblicks ist auch eine Analyse, wie viele der gefundenen Techniken nur für den jeweiligen APT gefunden wurden, d.h., für andere APTs nicht identifiziert wurden. Ist die Überschneidung zwischen den Techniken verschiedener APTs groß, kann das auf eine geringe Trennschärfe hinweisen.

Aus der aggregierten Darstellung wird im Client nicht ersichtlich, welche Techniken genau identifiziert wurden. Daher gibt es andererseits eine genaue Auflistung der Techniken, die für einen Fall identifiziert wurden. Basierend auf den Daten der MITRE-Datenbank werden zu jeder Technik konkrete Gegenmaßnahmen präsentiert. Diese kann der User nutzen, um zu versuchen, den Angriff zu bekämpfen.

5.4.2. Darstellung in einer Falldokumentationsdatei

Neben dieser Darstellung auf der Seite des Falles gibt es auch eine detailliertere Falldokumentation als JSON-Download, sollte eine höhere Granularität gewünscht sein, als die Falldarstellung bietet. Die Dokumentation wird beim Download stets neu generiert anhand des jüngsten Zustandes des Falls. Sie entspricht einer vorgegebenen Referenz-Struktur. Die Bestandteile dieser Struktur sind:

1. Dateiname der Dokumentation
2. Erstelldatum der Dokumentation
3. Falldaten:
    a) Fallname
    b) Erstelldatum
    c) Benutzerdaten:
        i. Username
    d) Aggregierte Auswertung nach APT:
        i. Anzahl erkannter Techniken gesamt
        ii. Anzahl der Techniken, die in sämtlichen Logfiles des Falls ausschließlich für den jeweiligen APT erkannt wurden
    e) Analysierte Logfiles:
        i. Informationen zum Logfile (Dateiname, Uploaddatum, Dateipfad, Inhalt als Liste von Logzeilen)

   ii. Analyse des Logfile sortiert nach APTs (erkannte Techniken und Anzahl der Indikatoren, d.h. Logzeilen, in denen die Technik für den APT erkannt wurde, je APT)

f) Gegenmaßnahmen:
   i. Allgemeine Beschreibung der jeweiligen Gegenmaßnahme
   ii. Liste von spezifischen Beschreibungen der Gegenmaßnahmen pro Technik, auf die die Gegenmaßnahme zutrifft

g) Feedback zu Gegenmaßnahmen:
   i. Ausgewählte Feedback-Option
   ii. Freitextfeedback

5.4.3. Feedback zu Gegenmaßnahmen

Zur laufenden Verbesserung der Usability ist die kontinuierliche Einholung des Feedback von Usern wichtig. Daher ist es dem User auch möglich, Feedback zu den vorgeschlagenen Gegenmaßnahmen zu geben. Erstens kann der User von im Prototyp vorgegebenen drei Optionen in einem Dropdown-Menü wählen („not helpful“, „helpful“, „partially helpful“). Er kann zweitens beliebiges zusätzliches Feedback in einem Freitextfeld absenden. Das Feedback kann verändert werden.

6. Implementierung

Dieses Kapitel beinhaltet die technischen Details der Umsetzung des in Abschnitt 5 erwähnten Designs. Es werden der verwendete Technologiestack, die Struktur des implementierten Clients, das Deployment und Nutzer-Interface sowie die Sicherheit des Clients beschrieben.

6.1. Verwendeter Technologiestack

In diesem Abschnitt wird der verwendete Technologiestack in der Implementierung des Clients behandelt.

6.1.1. Programmiersprachen und Entwicklungsumgebung

Der Client wurde überwiegend in *Python 3.8* programmiert. Python ist eine gut dokumentierte bzw. verbreitete Sprache und das containerisierte Deployment ist erprobt (siehe Abschnitt 6.2). Neben Python gibt es auch einen Anteil an *HTML* und *Jinja2*. Bei Jinja2 handelt es sich um die integrierte Templatesprache des Web-Framework *Flask* (siehe Abschnitt 6.1.2). Beide Sprachen werden für die Implementierung des User-Interface verwendet.

Neben den genannten Sprachen ist auch die von *YARA* bereitgestellte Syntax wesentlich für die Implementierung. Mit der YARA-Syntax können Erkennungsregeln für böswillige Aktivität verfasst werden. YARA, ein verbreitetes Tool für Malwareforschung, wird in Abschnitt 6.1.2 genauer beschrieben.

Vereinzelt kommen in den für das Deployment relevanten Implementierungsbestandteilen auch Shell-Befehle vor (siehe 6.2).

Als IDE wurde *PyCharm*[Jet] verwendet. Sie ist spezifisch auf das Implementieren von Python-Projekten ausgelegt.

6.1.2. Libraries und Frameworks

Folgende Libraries und Frameworks wurden in der Implementierung der Python-Anwendung verwendet:

- **Flask, Jinja2:** Mit dem verbreiteten Webframework *Flask*[Pala] wird der Prototyp als Python-Webanwendung implementiert. Als Webanwendung ist der Prototyp betriebssystemunabhängig und kann lokal laufen. Das Deployment kann künftig auch auf eigens bereitgestellten Servern vorgenommen werden. Damit kann auch die Leistung der Webanwendung skaliert

werden und z.B. der performante Zugriff mehrerer User gleichzeitig und parallele Prozesse ermöglicht werden. *Jinja2*[Palb] ist die zu Flask gehörige Template-Sprache. Sie ermöglicht es, dynamische Inhalte, die von den View-Methoden der Anwendung bereitgestellt werden, im User-Interface anzuzeigen. Zusätzlich kann man mit Jinja2 Vererbungsbeziehungen zwischen Templates definieren. So muss man z.B. eine gemeinsame Navigationsleiste für verschiedene Templates nur einmal definieren.

- **flask-mongoengine, mongoengine, pymongo:** Bei *mongoengine*[Mond] handelt es sich um einen Object-Document-Mapper für die Arbeit mit einer MongoDB-Datenbank in einer Python-Anwendung. Mongoengine erleichtert die Arbeit mit MongoDB-Datenbanken in der Anwendung, indem man mit Datenbankeinträgen objektorientiert mit regulärer Python-Syntax arbeiten kann. Das Schema einer Collection in der MongoDB wird mit den von mongoengine bereitgestellten, auf MongoDB abgestimmten Felder-Klassen definiert. Der Username einer UserAccount-Instanz wird bspw. definiert als vom Typ „StringField“, wie von mongoengine bereitgestellt, anstatt einfach als String-Objekt, wie von Python standardmäßig definiert. Der Vorteil liegt darin, dass mongoengine selbtständig Integritätschecks durchführt, damit garantiert wird, dass die eingegebenen Daten in der MongoDB-Instanz in dieser Form erlaubt sind. Bei Zuweisung eines Wertes zu einem „URLField“ wird z.B. automatisch geprüft, ob es sich bei diesem Wert um eine gültige URL handelt. *flask-mongoengine*[Palc] stellt einen Wrapper um mongoengine bereit und bezieht die Verbindungsdaten der MongoDB-Instanz aus der Konfiguration der Flask-App. *pymongo*[Monc] ist der mongoengine zugrundeliegende MongoDB-Treiber für Python.

- **yara-python:** *YARA*[Yar] ist ein verbreitetes Framework in der Malware-Forschung. Es stellt eine Syntax zur Definition von Angriffsmustern in Form von Erkennungsregeln bereit. *yara-python*[Gitb] ist ein Interface zur Verwendung von YARA in einer Python-Anwendung. Man kann damit die YARA-Regeln kompilieren und Aktivitätslogs (z.B. Netzwerklogs), die möglicherweise einen Angriff abbilden, auf Übereinstimmungen mit den kompilierten Regeln prüfen.

- **uWSGI:** *uWSGI*[Read] ist eine Implementierung eines WSGI-Servers. WSGI-Server sind erforderlich, um Python-Anwendungen auf Webservern zu betreiben.[Wik] uWSGI ist somit Teil des Deployment. Es ist die Schnittstelle zwischen der Anwendung und dem Web. Details zum Deployment der Anwendung finden sich in Abschnitt 6.2.

- **stix2:** *STIX2*[Reac] ist eine Library, die Methoden bietet, um mit Daten im STIX2-Format zu arbeiten. STIX2 ist ein Standard für die Dokumentation und den Austausch von Informationen aus dem Bereich der Cyber Threat Intelligence.[OAS] Diese Informationen können mit den von STIX2

gebotenen Standard-Domain-Model als JSON-Datei dokumentiert werden. Für den Prototyp ist dies relevant, weil die „MITRE ATT&CK Enterprise 9.0“-Datenbank in Form eines JSON-Objektes vorliegt, das in einem leicht adaptierten STIX2-Format geschrieben ist.[Gita] Um die Informationen aus der Datenbank zu extrahieren und sie vom STIX2-Model in das Domain-Model des Prototypen zu überführen, ist die Benutzung der stix2-Library nötig.

- **Flask-Login:** *flask-login*[Reaa] erweitert Flask um grundlegende Methoden zum Session-Management. Es ist zentral für die Zugriffskontrolle im Prototypen durch vorgeschriebenen User-Login.

- **Flask-Markdown:** Die Beschreibungstexte der MITRE-Datenbank für Techniques sind im Datenbank-JSON mittels Markdown formatiert. *Flask-Markdown*[Reab] erweitert Jinja2 um eine Anweisung, dass ein bestimmter String in Markdown zu formatieren ist, anstatt wie standardmäßig in HTML. Es sorgt für die korrekte Darstellung von Markdown im User-Interface.

- **matplotlib, numpy:** *numpy*[Num] ist eine Python-Library für numerische Berechnungen und lineare Algebra. *matplotlib*[Thea] ist eine Library, um Auswertungen zu visualisieren. Mittels matplotlib werden Plots über die Analyseergebnisse des Prototyps visualisiert. Dabei wird matplotlib unterstützt von Hilfsfunktionen aus numpy. Konkret wird mit den zwei Libraries auf der Seite eines Falls im User-Interface grafisch dargestellt, wie viele Techniken für das jeweilige APT erkannt wurden und wie viele davon ausschließlich für das jeweilige APT erkannt wurden.

- **Flask-WTF, WTForms:** *WTForms*[WTFb] ist eine Python-Library zum Erstellen von Formularen. Bei *Flask-WTF*[WTFa] handelt es sich um einen für die Nutzung mit Flask ausgelegten Wrapper um die WTForms-Library. Flask-WTF bietet zusätzliche Funktionen, etwa Sicherheitsfeatures wie CSRF-Tokens und eine Uploadfunktionalität, die auf Flask-Apps abgestimmt ist. Flask-WTF wird verwendet, um Inputformulare im User-Interface zu definieren (z.B. den Logfileupload oder die Abgabe von Feedback zu Gegenmaßnahmen eines Falles).

Darüber hinaus gibt es eine Reihe an kleineren Libraries und Frameworks sowie einige Dependencies. Eine vollständige Übersicht über die installierten Libraries und Frameworks findet sich in der Datei *requirements.txt*. Die Libraries und Frameworks sind mit Versionsnummern versehen, um Kompatibilität untereinander zu garantieren.

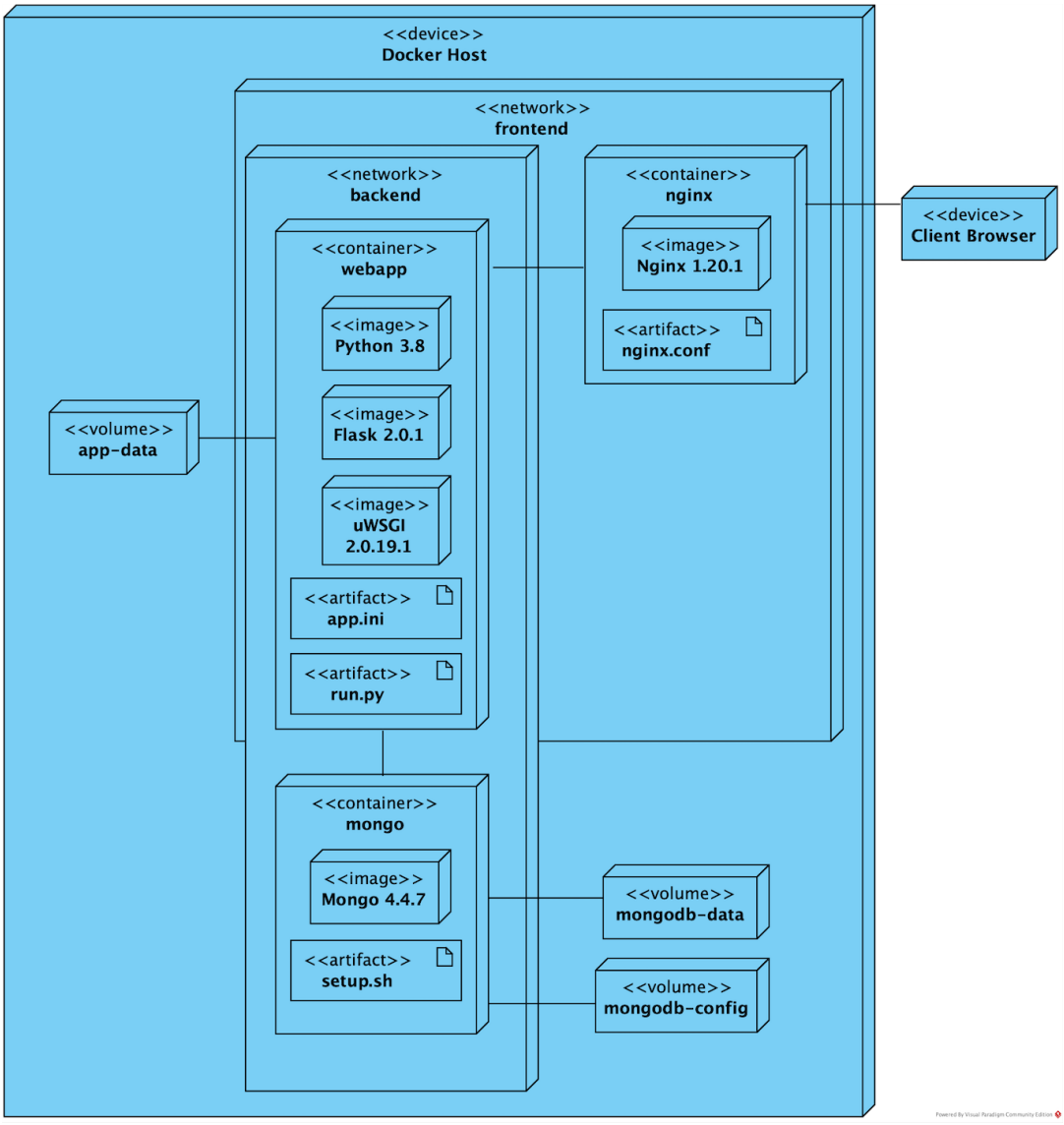


Abbildung 6.1.: Deploymentdiagramm für die lokale Ausführung des Prototyps als containerisierte Webanwendung

6.2. Architektur und Deployment

Der Client-Prototyp ist als Webapplikation konzipiert und implementiert. Ziel ist einerseits Plattformunabhängigkeit, andererseits Skalierbarkeit. Beim Prototyp handelt es sich um eine Container-Applikation[Doce], wiederum aus Gründen der Plattformunabhängigkeit. Als Container-Tool wurde Docker[Docb] ausgewählt, da es weit verbreitet und gut dokumentiert ist.[Diga]

Das Deployment für den Prototypen findet lokal statt. Die Webanwendung wird also im Rahmen der Entwicklung des Prototypen nur lokal gehostet. Der Zugriff ist nach Starten der Docker-Anwendung über `localhost:80` möglich. Ein Deployment auf einem im Internet zugänglichen Webserver würde den Scope dieser Arbeit übersteigen und bedürfte zusätzlicher Ressourcen zur Sicherung von unbefugtem Zugriff.

Auf dem Host-PC wird eine Docker-Anwendung ausgeführt, die aus verschiedenen Containern besteht – siehe Abb. 6.1 für ein Deployment-Diagramm der Anwendung. Jedem der Container entspricht ein Dockerfile, das den konkreten Aufbau des Containers bzw. Services definiert. Diese Dockerfiles werden über eine Datei `docker-compose.yml`[Docc] aufgerufen. Im Compose-File werden die einzelnen Services miteinander in Verbindung gesetzt und deren Netzwerkeigenschaften verwaltet (z.B. offene Ports zur Kommunikation über das Netzwerk). Darüber hinaus werden im Compose-File auch relevante Umgebungsvariablen definiert (die konkreten Werte der Variablen werden aus Sicherheitsgründen über ein `.env`-File bereitgestellt und stehen nicht „hard-coded“ im Compose-File).

Die folgenden Services existieren als Container:

- **webapp**: Dieser Service umfasst den eigentlichen Webclient als Python-Anwendung mitsamt seiner Business- und Darstellungslogik. Zudem enthält er das uWSGI-Package mitsamt der Konfigurationsdatei `app.ini`. WSGI (Web Server Gateway Interface)[Med] ist ein Python-Standard, der die Kommunikationsschnittstelle zwischen einer Python-Webanwendung und einem Webserver, der diese bereitstellt (in diesem Fall NGINX, siehe unten), definiert.[Ski] Es gibt verschiedene Implementierungen eines WSGI-App-Server – die Wahl fiel hierbei auf uWSGI, da es sich um eine gut dokumentierte Implementierung handelt, insbesondere im Zusammenspiel mit dem NGINX-Web-Server.[Digb]
  Das Volume `app-data`, das in der Docker Engine bereitgestellt wird, enthält den Quellcode, das Python-Image mitsamt Erweiterungen sowie die uWSGI-Konfiguration. Flask (als das Web-Framework der App) und uWSGI (als der WSGI-Server) wurden im Diagramm gesondert aus einer Vielzahl anderer installierter Python-Packages (siehe 6.1.2) hervorgehoben, da sie relevant für das Deployment sind: Sie ermöglichen erst das Deployment der Anwendung als Webanwendung.
  Was die interne Kommunikation des webapp-Service mit den anderen Services im Docker-Host betrifft, ist der Service in beide bestehenden

Docker-internen Brückennetzwerke eingebunden (`backend` und `frontend`), da er sowohl mit dem `nginx`-Service (ausschließlich im `frontend`-Netzwerk) als auch mit dem `mongo`-Service (ausschließlich im `backend`-Netzwerk) kommunizieren muss. Zweck dieser Architektur ist, dass ein Service nur mit denjenigen Services kommunizieren kann, die er für sein eigenes Funktionieren braucht. Das reduziert die Anzahl der Kommunikationswege zwischen Services und damit auch mögliche Sicherheitsrisiken, da bei einem Angriff auf einem Service in einem bestimmten Netzwerk nur dieses Netzwerk direkt betroffen wäre.[Teca]
Das Dockerfile dieses Services bezieht ein Python-Image und legt einen Nutzer mit eingeschränkten Berechtigungen an, welcher den Container ausführen soll. Es installiert anschließend die erforderlichen Python-Packages für die Anwendung mittels `pip` und kopiert den Quellcode der Anwendung in das Image. Für die in diesem Schritt kopierten Dateien werden eingeschränkte Berechtigungen eingestellt. Zum Schluss wird der uWSGI-Server gestartet, über welchen die Prototyp-Webanwendung mit dem nginx-Service (und darüber mit dem Enduser) kommuniziert.

- **nginx**: Der nginx-Service ist mit dem webapp-Service über das frontend-Netzwerk verbunden. Er stellt den Prototyp-Client für externen Zugriff bereit. NGINX ist ein Web Server und wird wie im Fall des Prototyp-Clients häufig auch als „reverse proxy“[Kin] verwendet, z.B. im Zusammenspiel mit einem uWSGI-Server.[Digb] Bei NGINX handelt es sich um einen erprobten Web Server, der einige Funktionalitäten besitzt, die uWSGI nicht besitzt bzw. im Allgemeinen effizienter arbeitet und mit hohen Zugriffslasten umgehen kann.[Ser] Daher ist ein solcher Aufbau – NGINX als reverse proxy für einen uWSGI-Server – üblich.[Ine] Dieser Service besteht aus dem NGINX-Image selbst sowie aus einer Konfigurationsdatei, in der einerseits die vom uWSGI-Server für den NGINX-Server bereitgestellte Schnittstelle (somit die Schnittstelle zum webapp-Service), andererseits die vom NGINX-Server für den User bereitgestellte Schnittstelle definiert sind.
  Über das Dockerfile des Services wird das Nginx-Image bezogen und die Standardkonfigurationsdatei für NGINX durch die Konfigurationsdatei des Prototyps ersetzt.

- **mongo**: Der mongo-Service ist mit dem webapp-Service über das backend-Netzwerk verbunden. Er enthält ein Mongo-Image sowie ein Setup-Skript, das bei Initialisierung ausgeführt wird. Dieses legt eine Arbeitsdatenbank an sowie einen Nutzer, der bestimmte Lese- und Schreibrechte in der Arbeitsdatenbank besitzt, und importiert die MITRE-Datenbank in die neu angelegte Datenbank. Dieser Service stellt die Schnittstelle bereit zur Mongo-Datenbank, in der die für das Model der Anwendung relevanten Datensätze abgelegt, gelesen, geschrieben und bearbeitet werden können. Der Datenbankinhalt ist abgelegt im Volume `mongodb-data`, damit er inklusive

aller Änderungen während der Laufzeit über die Laufzeit der Anwendung hinweg gespeichert wird. Das Volume `mongodb-config` wird bereitgestellt, um das Initialisierungsskript zu speichern, das über einen Dockerfile-Befehl in das Image kopiert wird.[Mata]
Im Dockerfile wird das Mongo-Image heruntergeladen und es werden die relevanten Anwendungsdaten (das Initialisierungsskript und die MITRE-Datenbank) in das Image kopiert.

6.3. Ausführung des Clients

Die Ausführung des Clients erfolgt im Prototyp-Stadium über „Docker Compose“. Die Datei `docker-compose.yml` definiert die einzelnen Services und das Zusammenspiel zwischen ihnen – etwa Kommunikationsmöglichkeiten unter ihnen oder eine Startreihenfolge, sodass z.B. der Datenbank-Service gestartet wird, bevor der Webanwendungs-Service bereitgestellt wird, der den Datenbank-Service für seine Funktionalität braucht. Bei Vorhandensein einer solchen Konfigurationsdatei kann man den Client mit dem Kommandozeilenbefehl `docker compose up` aus dem obersten Verzeichnis des Client-Quellcodes heraus ausführen. Die Anwendung wird nach Ausführung des Befehls hochgefahren, beim ersten Hochfahren gegebenenfalls zuvor initialisiert, und kann anschließend über den Webbrowser über die Adresse `localhost:80` erreicht werden. Grundvoraussetzung ist das Vorhandensein einer „Docker Engine“-Installation.[Docd] Nicht jede „Docker Compose“-Version ist mit jeder „Docker Engine“-Installation kompatibel. Eine Kompatibilitätsmatrix befindet sich auf der Website von Docker.[Doca] Der Client verwendet die Compose-Version 3.8. Getestet wurde die Verwendung der Compose-Datei mit der Engine-Version 20.10.8. Gemäß Docker sollten sämtliche Versionen ab 19.03.0 aufwärts kompatibel sein. Die Anwendung kann geschlossen werden durch Abbruch in der Kommandozeile mittels der Tastenkombination `Strg+C`, über die grafische Benutzeroberfläche im „Docker Desktop“-Client, soweit installiert, oder mit dem Befehl `docker compose stop`.

Eine Schritt-für-Schritt-Anleitung zur Ausführung der Anwendung befindet sich im der Anwendung beigelegten Readme-Dokument. Würde nach dem Prototyp-Stadium die Anwendung dem Endnutzer bereitgestellt werden, könnte man z.B. eine Bereitstellung über das Web andenken, sodass der Endnutzer lediglich eine Domain aufrufen muss und der Webserver im Hintergrund mit dem „Docker Compose“-Tool arbeitet, um die Anwendung bereitzustellen.

6.4. Aufbau und Struktur des Quellcodes

Der Quellcode, der sich zur Gänze im Ordner `ids_prototype` befindet, ist nach den einzelnen Services geordnet, aus denen der Client-Prototyp zusammengesetzt ist (siehe Abb. 6.2). Ebenfalls auf oberster Ebene befindet sich das `docker-`

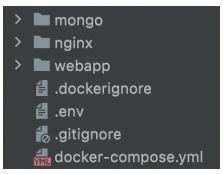


Abbildung 6.2.: Übersicht über Ordnerstruktur des Quellcodes (Screenshot aus PyCharm-IDE)

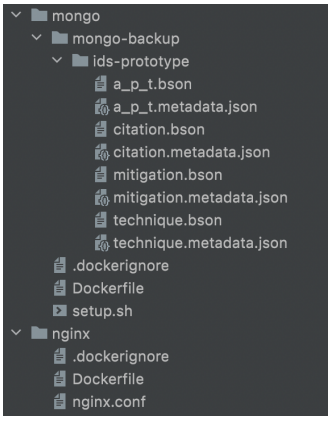


Abbildung 6.3.: Inhalt der Ordner „mongo“ und „nginx“ des Quellcodes für die gleichnamigen Services (Screenshot aus PyCharm-IDE)

`compose.yml`-File, das das Zusammenspiel der Services beschreibt (siehe Abschnitt 6.3). Das `.env`-File enthält die Werte von diversen Umgebungsvariablen, z.B. die Zugangsdaten, mit welchen die Mongo-Datenbank angelegt werden soll. Damit wird vermieden, dass diese an irgendeiner Stelle im Code „hard-coded“ vorkommen, was in Sachen Sicherheit eine best practice darstellt.[Mik]

In den Ordnern „mongo“ und „nginx“ für die gleichnamigen Services (siehe Abb. 6.3) befinden sich Konfigurationsdateien, die für die Erstellung und Definition der Services notwendig sind, so etwa Dockerfiles oder das `nginx.conf`-File (für Details zur Konfiguration der Services siehe 6.2). In „mongo“ befindet sich zusätzlich die „MITRE ATT&CK“-Datenbank, die von der Anwendung verwendet wird.

Im Ordner „webapp“ befindet sich der eigentliche Code der Webanwendung, d.h. der Code für die Python-Flask-Webanwendung und für die Darstellungs- sowie Businesslogik (siehe Abb. 6.4). Auf der obersten Ebene des Ordners befinden sich diverse Dateien, die für das Deployment der Webapp als Docker-Service relevant sind. Darunter fallen etwa die uWSGI-Konfigurationsdatei `app.ini` und das `run.py`-Modul, das von uWSGI zum Start der App aufgerufen wird. `config.py` definiert von der Anwendung zu beziehende Konfigurationsvariablen, deren Werte als Umgebungsvariablen aus der Docker-Umgebung entnommen werden. `requirements.txt` enthält die für die Anwendung erforderlichen Python-Packages. Auch ein Dockerfile ist vorhanden, das von Docker Compose verwendet wird, um den webapp-Service zu initialisieren.

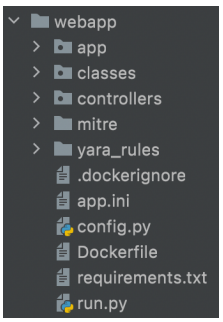


Abbildung 6.4.: Inhalt des Ordners „webapp“ des Quellcodes für den gleichnamigen Service (Screenshot aus PyCharm-IDE)

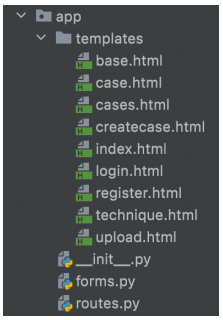


Abbildung 6.5.: Inhalt des Ordners „app“ des Quellcodes als Unterordner des „webapp“-Ordners (Screenshot aus PyCharm-IDE)

Die Anwendung ist nach dem Model-View-Controller-Pattern konzipiert; so entsprechen die Ordner „classes“, „app“ und „controller“ jeweils Model, View und Controller.

Der „app“-Ordner (siehe Abb. 6.5) umfasst die Bestandteile, die für die Verwendung des Flask-Webframework relevant sind. So wird etwa in `__init__.py` die Webanwendung an sich initialisiert und konfiguriert, einschließlich z.B. des Datenbankzugangs. `routes.py` enthält die View-Methoden. Eine Methode ist beispielhaft abgebildet in Abb. 6.6. Sie illustriert die Verwendung der von Flask bereitgestellten Python-Decorators wie `@app.route('/')`, die definieren, über welchen URL-Aufruf die View-Methode aktiviert wird sowie die Flask-Methode `render_template`, die ein Jinja2-/HTML-Template für die Darstellung im Browser rendert. Sämtliche Templates befinden sich im „templates“-Ordner. Das Template `base.html` ist dabei ein Grundtemplate, von dem die anderen Templates z.B. die Navigationsleiste oder gemeinsame HTML-Metadaten erben über die von Jinja2 bereitgestellte Vererbungsfunktionalität. Auch das `forms.py`-Modul ist Teil der View-Funktionalität, da es Klassen für verschiedene Eingabeformulare definiert, die durch Methoden in `routes.py` instanziiert und dargestellt werden sowie deren Inputs verwertet mithilfe von Instanzen der Klassen im „controllers“-Ordner.

Die Ordner „controllers“ und „classes“ (siehe Abb. 6.7) enthalten ausschließlich die Controller- und Modell-Klassen der Anwendung. Das Modell entspricht dabei

```python
@app.route('/')
@app.route('/index')
@login_required
def index():
    ...
    return render_template('index.html', title='Home')
```

Abbildung 6.6.: Beispiel für eine View Methode

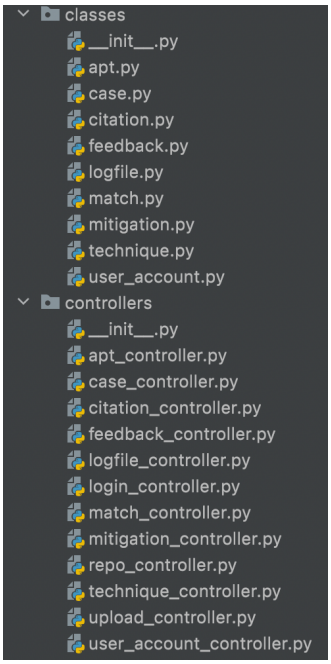


Abbildung 6.7.: Inhalt der Ordner „controllers“ und „classes“ des Quellcodes als Unterordner des „webapp“-Ordners (Screenshot aus PyCharm-IDE)

im Wesentlichen dem Modell, das die „MITRE ATT&CK“-Datenbank für die Analyse und Dokumentation von Cyberthreats bereitstellt, z.B. in Form der `APT-`, `Technique-` oder `Mitigation`-Klassen. Damit wird auch ein gemeinsames Interface für künftige allfällge Erweiterungen des Clients bereitgestellt. Jede Form der Detektion von Cyberangriffen, die künftig implementiert wird, hat ihre Resultate in Form des Models bereitzustellen.

In Abb. 6.8 werden die Inhalte der Ordner „mitre“ und „yara_rules“ dargestellt. Beim Inhalt des „mitre“-Ordners handelt es sich um eine Kopie des von MITRE bereitgestellten „MITRE ATT&CK“-Repositorys aus Gründen der Reproduzierbarkeit. Das Repository in der Version 9.0 wird von MITRE auf GitHub zum Download angeboten.[Mit] Im File `apt_rules.yar` sind die Erkennungsregeln gemäß der Syntax des YARA-Cyberangriffserkennungstools definiert. Jede der Erkennungsregeln hat einen Bezug zum MITRE-Model, das im Prototyp übernommen wurde. So ist z.B. für jede der Regeln definiert, auf welchen APT sie als Urheber deutet oder welcher Angriffstechnik sie zuzuordnen ist.

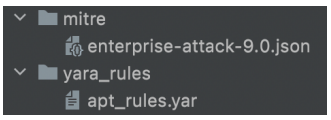


Abbildung 6.8.: Inhalt der Ordner „mitre“ und „yara_rules“ des Quellcodes als Unterordner des „webapp“-Ordners (Screenshot aus PyCharm-IDE)

6.5. Benutzeroberfläche

Der Schwerpunkt der Prototyp-Entwicklung lag auf der Funktionalität und nicht auf der Gestaltung der Benutzeroberfläche. Auf dem Weg zu einer möglichen Produktentwicklung könnte ihre Ausgestaltung ein wesentlicher Teil künftiger Entwicklungen sein (siehe dazu Kapitel **??**).

Auch wenn das Design der Benutzeroberfläche nicht im Vordergrund stand, wurde dennoch Wert gelegt auf Übersichtlichkeit und Verständlichkeit. Die Idee des Nutzerinterfaces war, die für den Prototyp entwickelten Use Cases abzubilden und deren Umsetzung so einfach und intuitiv wie möglich zu machen.

Dies beginnt bei der Navigationsleiste (siehe Abb. 6.9), die auf jeder Seite des Client-Prototyps gleich ist, wie ermöglicht durch die Vererbungsfunktionen der Jinja2-Template-Engine. In der Navigationsleiste werden die einzelnen Use Cases abgebildet.

Gestalterisch besonderer Augenmerk wurde gelegt auf die Seite zur Darstellung eines eigenen Cases (siehe Abb. 6.10, Abb. 6.11). Diese ist inhaltlich die komplexeste Seite der Anwendung, da sie die meisten Informationen umfasst. Der Einstieg in die Darstellung ist die Zusammenfassung der vom Prototyp durchgeführten Analyse sowohl textlich als auch visuell in Form einer Balkengrafik. Anschließend werden die Gegenmaßnahmen zu den identifizierten Attacken aufgelistet, mit der Möglichkeit, jede einzelne mit einer Bewertung sowohl in Form von Freitext als auch in Form einer Skala („Helpful'", „Not helpful“, „Partially helpful“) zu versehen.

Kommt es im Rahmen der Verwendung des Clients zu Fehlerfällen, erhält der Nutzer eine aussagekräftige Rückmeldung, wie beispielhaft abgebildet in Abb. 6.12.

Abbildung 6.9.: Navigationsleiste der Anwendung für einen eingeloggten User

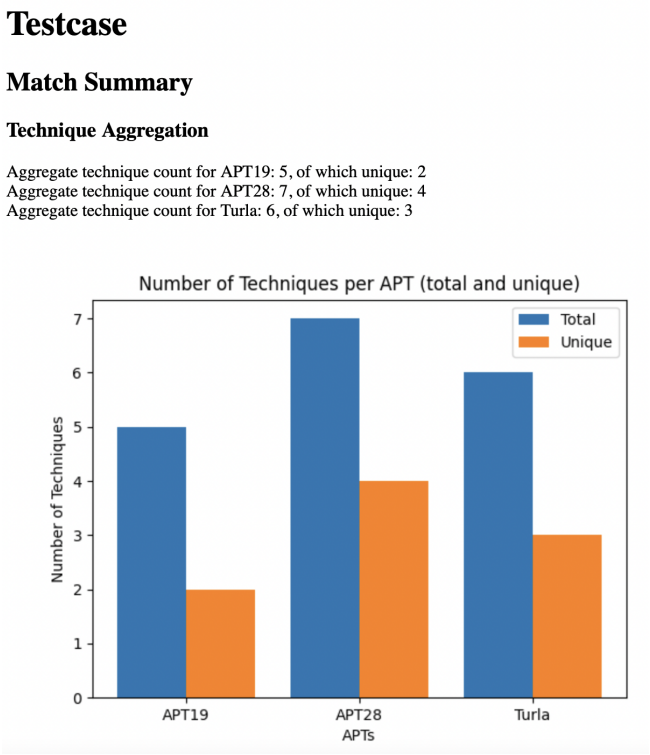


Abbildung 6.10.: Erster Teil der Falldarstellung: Auswertungsergebnis

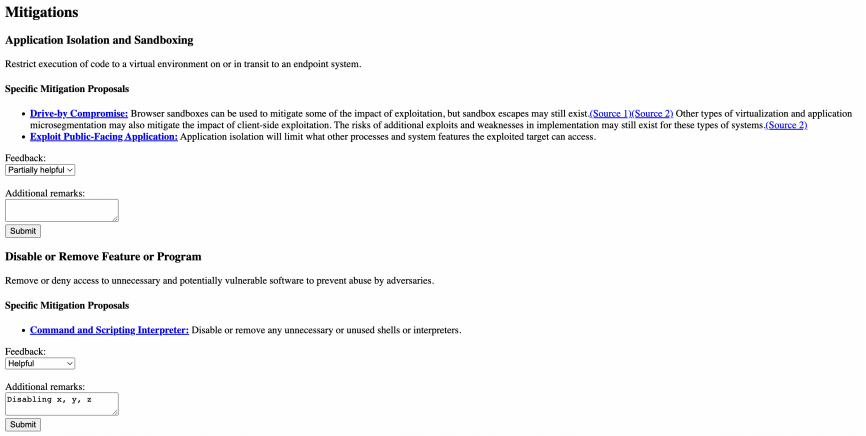


Abbildung 6.11.: Zweiter Teil der Falldarstellung: Mögliche Gegenmaßnahmen zu identifizierten Angriffen

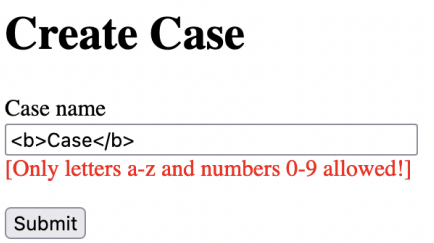


Abbildung 6.12.: Fehlermeldung bei Eingabe eines unerlaubten Fallnamens (dieser darf aus Sicherheitsgründen keine Sonderzeichen enthalten)

6.6. Sicherheit der Anwendung

Bei der Entwicklung des Client-Prototypes wurden best practices der Anwendungssicherheit berücksichtigt, insbesondere was jene Aspekte betrifft, die von den Requirements 8 und 17 festgelegt wurden (siehe Abschnitt 4.2).

Requirement 8 verlangt, dass der Zugriff auf Userdaten nur für den befugten Nutzer möglich ist. Dies wurde über die Implementierung des Tools `flask-login` gewährleistet, das die erforderlichen Funktionalitäten für Session-Management in Flask-Anwendungen integriert. Indem View-Methoden mit dem Decorator `@login_required` versehen werden, sind sie nur für eingeloggte User sichtbar. Nicht eingeloggte User werden auf die Login-Seite der Anwendung verwiesen.

Über das Session-Management ist es auch möglich, im Code die Daten über den aktuell eingeloggten User abzurufen. Dies ist relevant für eine zusätzliche Ebene der Zugriffskontrolle: So kann sichergestellt werden, dass es nicht einfach reicht, eingeloggt zu sein, um bestimmte Seiten zu besuchen, sondern dass ein bestimmter User eingeloggt sein muss. Möchte etwa ein User die Dokumentation eines Falls herunterladen, wird geprüft, ob sein Username enthalten ist im Pfad der Dokumentation (die Falldokumentationen der Fälle eines Users sind in einem nach dem User benannten Verzeichnis abgelegt).[Miga] Entsprechend den best practices der Softwareentwicklung wird das Passwort eines Users ausschließlich gehasht gespeichert und scheint zu keinem Zeitpunkt im Klartext auf.[Migb]

Requirement 17 verlangt Absicherung gegen Injection-Attacken[OWAa]. Einer Injection wird vorgebeugt über verschiedene Maßnahmen: Erstens wird der mögliche Input durch externe Quellen beschränkt. Dies geschieht z.B. durch Validierungen von Formularfeldern. Ein Beispiel ist die Beschränkung von Dateigrößen oder -endungen (über eine Allowlist für `.log` und `.txt`) bei Uploads sowie eine Blocklist für Input mit Zeichen, die verwendet werden könnten, um einen Injection-Angriff durchzuführen. Konkret wird z.B. die Verwendung der Zeichen `\$` und `\{` untersagt, da diese eine MongoDB-Injection ermöglichen könnten.[Mona]

Zweitens wird zugelassener Input weiteren Prüfungen unterzogen, etwa durch das Sanitizing der Namen hochgeladener Logdateien.[Pale] Ein Angreifer könnte sich über entsprechend gewählte Dateinamen theoretisch Zugriff auf Dateien

verschaffen, auf die er nicht zugreifen soll.[OWAb] Dies wird somit unterbunden.

Drittens wird der Output des Clients im Rendern der Seiten durch Flask und seine Template-Engine Jinja2 automatisch escaped.[Pald] Wenn also potentiell schadhafter Input in der Datenbank abgelegt wird und aus dieser bezogen wird, wird er nicht ausgeführt.

7. Evaluierung

Um den Client zu evaluieren, sollen Angriffe der behandelten APTs möglichst realitätsnah nachgebildet werden. Grundlage dieser Nachbildung sind *PCAP*-Dateien, die durch das Analysetool *Wireshark* [Wir] generiert wurden. In diesen Dateien wird der gesamte Netzwerkpaketverkehr über einen bestimmten Zeitraum erfasst. Diese PCAP-Dateien werden anschließend mit dem Analysetool *Suricata* [Thej] auf verdächtige Aktivitäten geprüft. Ergebnis eines solchen Scans ist ein Suricata-Log im *Fast*-Format, auf welches der Client ausgerichtet ist. Dieses Log kann verändert werden, indem zusätzliche Logeinträge eingefügt werden.

Indem diese Logeinträge passend gewählt werden, kann im Suricata-Log ein Angriff durch einen bestimmten APT nachgestellt werden. Die derart veränderten Logs werden schließlich im Client zur Analyse hochgeladen. Das Kriterium zur Evaluierung ist, ob der Client einen Angriff eines bestimmten APT als solchen erkennt und wie trennscharf er den simulierten APT als Angreifer identifizieren kann.

Die PCAP-Dateien, die die Grundlage für die Evaluierung darstellen, wurden gezielt generiert durch das Besuchen diverser Webseiten unterschiedlicher Kategorien (wie z.B. Nachrichten, Online-Shopping, Streaming etc.) über einen bestimmten Zeitraum auf einem Windows-10-PC. Während dieser Besuche ist die Wireshark-Paketerfassung aktiv. In chronologischer Abfolge wurden folgende PCAP-Dateien generiert:

- **Misc_1**: am 05.11.2022 zwischen 14:29:34 und 15:37:07.
- **Misc_2**: am 06.11.2022 zwischen 14:41:44 und 15:07:34.
- **Misc_3**: am 06.11.2022 zwischen 15:17:34 und 15:55:42.
- **Misc_4**: am 12.11.2022 zwischen 11:01:22 und 11:40:37.
- **Misc_5**: am 12.11.2022 zwischen 12:03:36 und 12:32:31.

7.1. Tests des Prototyps

Die genannten PCAP-Dateien wurden mittels Suricata analysiert, mit dem Resultat eines Logfiles im Fast-Format mit dem Dateinamen *fast_[PCAP-Dateiname]*. Wie zu erwarten war, zeigen diese Logfiles keine verdächtige Aktivität und auch eine Analyse durch den Client-Prototypen ergab keine Treffer, die auf einen

Angriff durch eines der berücksichtigten APTs hinweisen würden, womit ein falsch-positives Ergebnis ausgeschlossen werden kann. Im Anschluss wurden, wie eingangs beschrieben, die Fast-Logdateien verändert, sodass sie in ihrer Gesamtheit einen Angriff durch einen bestimmten APT nachbilden. Ob erstens der Angriff und zweitens der Angreifer (der APT) durch den Client-Prototypen erkannt werden können, ist das Kriterium der Evaluierung.

7.1.1. Test 1: Turla

Für den ersten Test wurden die Fast-Logdateien so bearbeitet, dass ein Turla-Angriff nachgestellt wird. Die einzelnen Phasen eines Turla-Angriffes sind beschrieben unter [Thef]. Für einen Teil dieser Phasen eines Turla-Angriffes werden zu diesen gehörende ausgewählte Techniken vom Client-Prototypen im Rahmen definierter YARA-Regeln erfasst.

In den Fast-Logdateien werden diesen Techniken entsprechende Suricata-Logeinträge ergänzt (siehe Appendix A für die konkret ergänzten Einträge). Teilweise handelt es sich bei diesen Logeinträgen um Logeinträge, die spezifisch einem Turla-Angriff mittels der Technik entsprechen. Teilweise handelt es sich bei den Logeinträgen auch um Logeinträge, die einem Angriff mittels der Technik entsprechen, der ein Turla-Angriff sein kann, aber nicht muss. Ein solcher Logeintrag weist an sich nicht spezifisch auf konkrete Turla-Urheberschaft hin, was für die meisten Techniken technisch sehr schwierig nachzuweisen wäre (häufig ist dies nur möglich z.B. wenn die verwendete Ziel-Domain für einen Angriff bekanntermaßen zu Turla gehört, ein Ausmaß an Detail, das Suricata für viele Regeln und Techniken nicht bietet). Er weist stattdessen nur auf eine von Turla bekanntermaßen verwendete Technik hin und ist somit ein Indiz für die Turla-Urheberschaft einer Angriffsaktivität.

Die Zuordnung von Phasen und Techniken einerseits, den Logdateien andererseits, lautet wie folgt:

1. *fast_Misc_1*: Phase „Initial Access“, Technik „Drive-by Compromise“.
2. *fast_Misc_2*: Phase „Lateral Movement“, Technik „Lateral Tool Transfer“.
3. *fast_Misc_3*: Phase „Lateral Movement“, Techniken „Remote Services: SMB/Windows Admin Shares“ sowie „Lateral Tool Transfer“ (Indikator für beide Techniken im ergänzten Logeintrag).
4. *fast_Misc_4*: Phase „Command and Control“, Technik „Application Layer Protocol: Web Protocols“.
5. *fast_Misc_5*: Phase „Exfiltration“, Technik „Exfiltration Over Web Service: Exfiltration to Cloud Storage“.

Das Hochladen der derart veränderten Logs im Client ergibt, dass eine Technik in den Logs erkannt wurde, die APT19 zugeordnet werden kann, und fünf

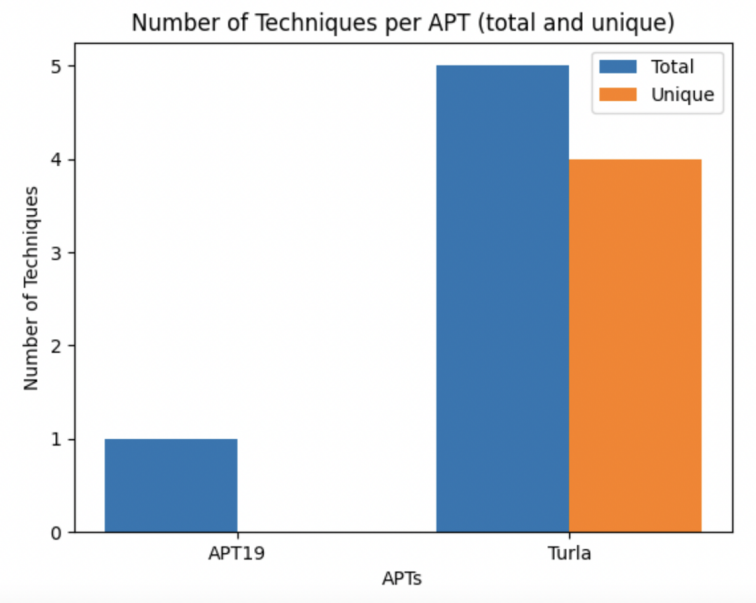


Abbildung 7.1.: Übersicht über identifizierte Techniken aus Test 1

Techniken, die Turla zugeordnet werden können. Der Indikator für die Technik, die APT19 zugeordnet werden kann, kann auch Turla zugeordnet werden. Weitere Überschneidungen gibt es in diesem Testdatensatz nicht – siehe Abb. 7.1.

Das Ergebnis zeigt somit eine gewisse Trennschärfe der Regeln, die der Client-Prototyp definiert und prüft, denn die Regeln sind in der Lage, bei wenigen Überschneidungen die erkannten Techniken jeweils nur bestimmten APTs zuzuordnen. Angesichts der Tatsache, dass es für Turla besonders viele spezifische Indikatoren im Vergleich zu APT19 und APT28 gibt, ist dieses Ergebnis folgerichtig. Zudem treten auf der Netzwerkebene viele Turla-Indikatoren auf, welche der gegenwärtig auf Suricata-Logs fokussierte Client-Prototyp erkennen kann (anders als z.B. Angriffsindikatoren, die auf der Ebene von PowerShell-Logs sichtbar werden, die von Suricata und vom Client daher aktuell nicht behandelt werden).

7.1.2. Test 2: APT19

Bei dem Test für APT19 wird analog zum Test für Turla vorgegangen. Die einzelnen Phasen eines APT19-Angriffes sind beschrieben unter [Thed]. Die Zuordnung der Phasen und ihrer Techniken zu den Logdateien lautet hier wie folgt (siehe auch hier Appendix A):

1. *fast_Misc_1*: Phase „Initial Access“, Technik „Phishing: Spearphishing Attachment“ sowie „Drive-by Compromise“ (Indikator für beide Techniken im ergänzten Logeintrag).

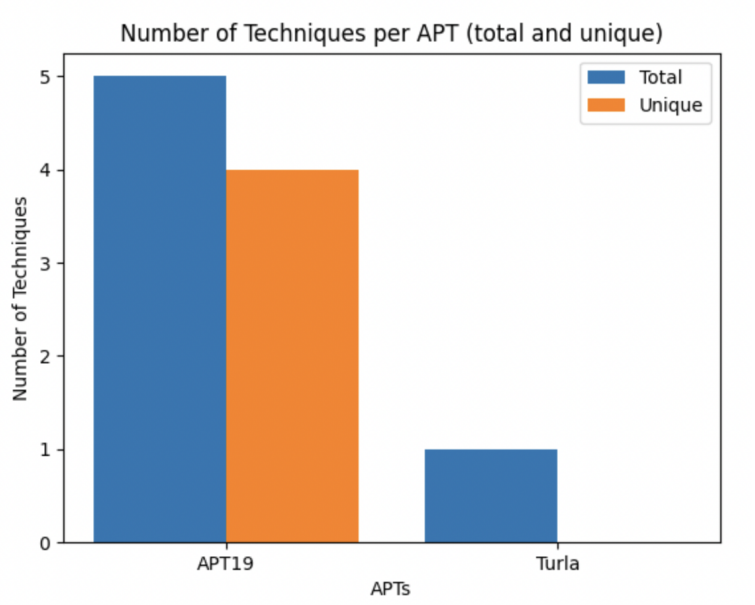


Abbildung 7.2.: Übersicht über identifizierte Techniken aus Test 2

2. *fast_Misc_2*: Phase „Execution“, Technik „Command and Scripting Interpreter“.

3. *fast_Misc_3*: Phase „Command and Control“, Techniken „Application Layer Protocol: Web Protocols“.

4. *fast_Misc_4*: Phase „Command and Control“, Technik „Data Encoding: Standard Encoding“.

5. *fast_Misc_5*: Phase „Command and Control“, Technik „Application Layer Protocol: Web Protocols“.

In der Analyse, die der Client-Prototyp die Logs einer Auswertung unterzieht, sind, bis auf eine, sämtliche fünf entdeckten Techniken ausschließlich APT19 zuordenbar – siehe Abb. 7.2. Lediglich eine Technik kann auch Turla zugeordnet werden. Das bedeutet, dass mit einer Auswahl aus den im Client-Prototyp definierten Regeln eine trennscharfe Zuordnung getätigt werden kann. Da die Logdateien selbst aus praktischen Gründen eine relativ kleine Größe haben, ist die Anzahl erkannter Indikatoren insgesamt auch relativ klein. Damit ist auch die Wahrscheinlichkeit geringer, dass es zu Überschneidungen mit Indikatoren für andere APTs kommt, als sie es in echten Fällen mit deutlich größeren Logs wohl wäre. Dazu kommt auch, dass das Volumen an Paketen in realen Fällen deutlich höher und diverser sein wird als in den simulierten Fällen. Dies gilt grundsätzlich auch für die Tests zu Turla und APT28.

7.1.3. Test 3: APT28

Der letzte Test anhand der veränderten Logdateien ist der Test für APT28. Auch hier wurden einzelne Phasen simuliert gemäß der Beschreibung unter [Thee]. Hier lautet die Zuordnung der Phasen und ihrer Techniken zu den Logfiles wie folgt (siehe auch Appendix A):

1. *fast_Misc_1*: Phase „Reconnaissance“, Technik „Active Scanning: Vulnerability Scanning“; zudem Phase „Initial Access“, Technik „Exploit Public-Facing Application“ (jeweils ein Logeintrag pro Technik).

2. *fast_Misc_2*: Phase „Command and Control“, Technik „Application Layer Protocol: Web Protocols“.

3. *fast_Misc_3*: Phase „Command and Control“, Technik „Application Layer Protocol: Mail Protocols“.

4. *fast_Misc_4*: Phase „Command and Control“, Techniken „Data Obfuscation: Junk Data“ sowie „Application Layer Protocol: Web Protocols“ (zwei Logeinträge).

5. *fast_Misc_5*: Phase „Command and Control“, Techniken „Encrypted Channel: Symmetric Cryptography“ sowie „Application Layer Protocol: Web Protocols“ (zwei Logeinträge – einer davon für beide Techniken, einer nur für „Encrypted Channel: Symmetric Cryptography“).

Die Analyse durch den Client-Prototypen ergibt eine deutliche Tendenz für APT28 bei sechs dafür identifizierten Techniken, siehe Abb. 7.3. Zugleich gibt es auch einen Treffer für Turla, darunter jedoch keinen, der nicht auch eine Technik für APT28 beschreiben würde. Das bildet die Tatsache ab, dass realistischerweise nie alle Angriffsaktivitäten exakt einem APT zuordenbar sein werden, sondern dass verdächtige Aktivitäten sich zwischen verschiedenen APTs auch gleichen können. Ein Beispiel ist etwa ein Indikator für ein verdächtiges HTTP-Paket, das helfen kann, einen Angriff festzustellen, für sich aber nicht Auskunft darüber gibt, welcher konkrete APT hinter einem Angriff steckt. Trotz mancher gemeinsamer Indikatoren vermag der Client-Prototyp dem simulierten Angriff zu entnehmen, dass es insgesamt einige APT28-spezifische Indikatoren gibt, die dafür sprechen, dass der Angriff konkret von APT28 kommt und eben nicht von einem anderen APT.

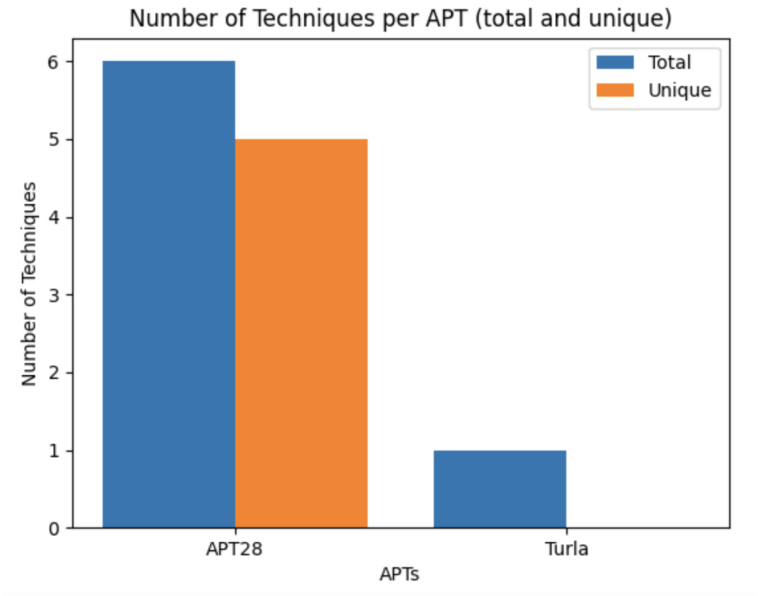


Abbildung 7.3.: Übersicht über identifizierte Techniken aus Test 3

7.2. Generelle Resultate der Evaluierung

Im Folgenden werden einige Learnings aus der gesamten Evaluierung beschrieben.

7.2.1. Granularität der Analyse

Die Analyse beruht im Hintergrund auf dem Erkennen von Indikatoren in einem hochgeladenen Logfile. Diese Indikatoren sind mindestens einer bestimmten Technik sowie mindestens einem bestimmten APT zugeordnet. Wenn also ein Indikator in einem Logfile identifiziert wird, wird aufgezeichnet, welchen APTs und welchen Techniken er zugehörig ist. Die Darstellung der Analyse jedoch wird nicht pro Logdatei vorgenommen, sondern über einen gesamten Fall, der beliebig viele Logfiles enthält. Für jeden Fall wird dem Nutzer dargestellt, welche Techniken für welchen APT gefunden wurden, und ob diese nur für ein APT identifiziert wurden oder für mehrere.

Es wird damit von der Analyse auf Ebene der Indikatoren abstrahiert und deren Resultate deutlich vereinfacht. Damit einher geht ein Verlust an Granularität bzw. Genauigkeit. Grund für diese Designentscheidung ist erstens, die Analyse als Einschätzungshilfe übersichtlich zu halten. Zweitens soll anhand einer genaueren Darstellung keine Genauigkeit der Analyse suggeriert werden, die im Client-Prototyp – als Prototyp – noch nicht existiert. Würde stärker auf die Daten aus der Analyse über die identifizierten Indikatoren zurückgegriffen werden, könnten z.B. die Indikatoren danach gewichtet werden, wie schwerwiegend die Angriffsaktivität ist, die sie anzeigen, oder danach, wie stark sie auf einen bestimmten APT hinweisen.

Um eine solche detailliertere Gewichtung seriöserweise vorzunehmen, wären jedoch mehr Tests, mehr Daten und mehr Indikatoren (siehe den folgenden Abschnitt 7.2.2) erforderlich gewesen. Diese hätte z.B. nicht zuletzt die Anzahl der Indikatoren pro APT sowie die Anzahl stärkerer oder schwächerer Indikatoren für einen APT berücksichtigen müssen, die auch angesichts der Beschränkung auf Logdateien von Netzwerkaktivitäten zwischen den APTs variiert. Diese Gewichtung hätte folglich in ihrer Komplexität den Rahmen dieser Arbeit überstiegen. Jedoch lassen sich aus diesem Learning Empfehlungen für eine Weiterentwicklung des Client-Prototypes ableiten – siehe Abschnitt 7.3.

7.2.2. Verfügbarkeit von Daten

Aufgrund beschränkter zeitlicher Ressourcen konnte nur ein begrenzter Ausschnitt eines APT-Angriffes berücksichtigt werden, konkret die Netzwerkaktivität eines APT im Rahmen eines Angriff, wie sie von Suricata auf Netzwerkpaketebene identifiziert werden kann. Andere Aktivitäten wie z.B. lokale Aktivität in Gestalt von Scripts für PowerShell, Änderungen in der Registry oder ähnliches können hierüber nur identifiziert werden, insoweit sie Auswirkungen auf Netzwerkebene haben. Zwar lässt sich angesichts der Ergebnisse der Evaluierung behaupten, dass die alleinige Betrachtung der Netzwerkaktivität durchaus in der Lage ist, Angriffe anzuzeigen. Je nach konkreter Ausgestaltung der Netzwerkaktivität ist es auch möglich, mit einiger Verlässlichkeit einzugrenzen, welcher APT am wahrscheinlichsten hinter einem Angriff steckt. Dennoch kann der Client-Prototyp oft nur allgemeine Indikatoren feststellen, weil Suricata für die berücksichtigten APTs nur begrenzt viele Indikatoren bietet, die auf der Netzwerkebene identifiziert werden können – insbesondere spezifischere Indikatoren sind auch abhängig vom konkreten APT eher selten.

Eine zweite Beschränkung ist, dass nur eine beschränkte Anzahl von APTs berücksichtigt wurde. Da der Client-Prototyp nur bestimmte APTs berücksichtigt, wird er klarerweise einen Angriff zumindest einem dieser APTs zuweisen, sofern er einen Angriff identifiziert. Die Möglichkeit, dass in Wahrheit eines der nicht berücksichtigten APTs hinter einem Angriff steckt, kann per Definition nicht berücksichtigt werden. Damit kann es zu falsch-positiven- (ein APT wird für einen anderen gehalten) und falsch-negativen- (es wird kein Angriff gefunden, da der angreifende APT nicht berücksichtigt wird) Ergebnissen kommen. Auch hier konnten aus Ressourcengründen nicht mehr APTs berücksichtigt werden, da diese zahlreich sind und eine teils sehr breit gefächerte Vorgehensweise aufweisen.

Drittens ist die Evaluierung an sich herausfordernd. Für diese sind PCAP-Dateien erforderlich, die einen entsprechenden Angriff abbilden. In Sandbox-Systemen könnte bei gegebener Malware versucht werden, derartige Angriffe zu simulieren, was jedoch angesichts der Sicherheitsrisiken solcher Simulationen die Ressourcen dieser Arbeit überstiegen hat. Daher war es notwendig, mit eigenhändig nachgestellten Logfiles von hypothetischen Angriffsfällen zu testen. Diese sind womöglich zu sehr darauf ausgerichtet, was der Client-Prototyp

erkennen kann, und täuschen damit eine bessere Performance vor, als gegenüber einem Angriff „in freier Wildbahn“ gegeben wäre.

7.2.3. Identifikation von Angriffen

Insgesamt vermag der Client-Prototyp verlässlich solche Angriffe zu erkennen, die spezifisch einem der berücksichtigten APTs entsprechen. Der Prototyp ist auch in der Lage, Indikatoren über den gesamten „Life Cycle“ (entsprechend der Mitre Phasen) eines Angriffs zu finden, jedenfalls soweit dieser auf Netzwerkebene abbildbar ist. Angesichts der Komplexität des Vorgehens von APTs und gleichzeitig begrenzter Ressourcen ist dies ein zufriedenstellendes Resultat.

7.3. Empfehlungen aus der Evaluierung

Die Resultate sind insgesamt plausibel und zufriedenstellend. Um die Qualität der Analyse weiter zu erhöhen, können zusammenfassend folgende Maßnahmen angedacht werden (bei genauer Abwägung der vorangehend ausgeführten Vor- und Nachteile, siehe 7.2):

- Erhöhung der Anzahl berücksichtigter APTs,
- Berücksichtigung weiterer möglicher Quellen für Logfiles zur Analyse; entsprechende Erweiterung der vom Client-Prototyp abgedeckten Indikatoren, Techniken und Angriffsphasen,
- Gewichtung von Indikatoren nach Stärke und/oder Aussagekraft für einen gegebenen APT oder eine gegebene Technik, Häufigkeit (pro Logdatei und/oder in einem Fall) sowie Schweregrad,
- Visualisierung auf Ebene von (gewichteten) Indikatoren pro Logdatei und/oder Fall, anstatt auf Ebene von Techniken pro Fall,
- Visualisierung der konkreten Logzeilen, in welchen Indikatoren vorkommen, sowie Visualisierung der Indikatoren,
- Ausweitung der Evaluierungsmöglichkeiten durch Analyse und Logging von Malware- oder APT-Aktivität in einer Sandbox-Umgebung.

Zusammenfassend ist die Frage der optimalen Darstellung der Analyseergebnisse komplex. Es hängt auch von der Quellen- und Indikatorenlage ab, welche Aussagen in der Darstellung getroffen werden können und mit welcher Sicherheit. Ein Kompromiss könnte auch eine Art zusätzliche „Expertendarstellung“ sein, in der die Rohdaten der Analyse umfassend abgebildet werden. Eine Annäherung an eine solche Darstellung sind die Falldokumentationen, die im JSON-Format bereitgestellt werden. Diese Dokumentationen können grundsätzlich noch um weitere Informationen erweitert werden.

8. Conclusio

Dieses finale Kapitel widmet sich einerseits der möglichen Weiterentwicklung des Prototyps und andererseits den zentralen Schlussfolgerungen, in denen die zugrundeliegende Fragestellung beantwortet wird.

8.1. Modell und Prototyp als Ausgangspunkte für Weiterentwicklung

Da das Ziel des Prototyps war, die Funktionsfähigkeit des Modells zu demonstrieren, ist die derzeitige Implementierung zwar offen und erweiterbar, aber im Moment noch beschränkt. In diesem Abschnitt werden daher Vorschläge zu künftigen Maßnahmen zur Weiterentwicklung des Client-Prototyps gesammelt. Diese hätten den Rahmen der vorliegenden Arbeit überschritten, geben aber mögliche Richtungen zur weiteren Arbeit am Prototypen an.

8.1.1. Interface

Ein Bereich der Weiterentwicklung betrifft das Interface. Unter Rückgriff auf CSS können die einzelnen Elemente des grafischen Interface optisch ansprechender gestaltet werden. Im Bereich der Datenvisualisierung kann auf dezidiert zu diesem Zweck entwickelte Tools zurückgegriffen werden, um Auswertungsgrafiken ebenfalls optisch ansprechender zu gestalten sowie interaktive Elemente (z.B. Filter von Matches nach APTs oder beliebige andere Funktionen) zu integrieren.[Nik]

Weitere Verbesserungen der Usability sind möglich, etwa die Möglichkeit zum Upload mehrerer Logs zugleich oder eine Seite zur Verwaltung der Daten des eigenen Nutzerkontos. Auch die Möglichkeit zur Löschung hochgeladener Daten, erstellten Fällen u.a. kann in einer künftigen Entwicklungsphase gegeben werden. Generell werden Möglichkeiten und Richtung der Verbesserung des Interfaces beeinflusst durch jene Änderungen, die in den anderen Abschnitten des Ausblicks beschrieben werden – bspw. Verbesserungen in der Fallevaluierung durch den User, wie beschrieben in Abschnitt 8.1.4. Eine entsprechende Erweiterung des Funktionsumfangs zöge auch die Frage einer ansprechenden wie funktionalen Darstellung nach sich.

8.1.2. Inputquellen

Gegenwärtig ist der Regelsatz zum Auswerten von hochgeladenen Logs in zweifacher Hinsicht eingegrenzt.

Erstens war die Entwicklung des Prototyps im Scope fokussiert auf Erkennungsregeln zu drei bestimmten APTs. Es gibt eine Vielzahl weiterer APTs, zu denen noch Erkennungsregeln entworfen werden könnten. Dies bedeutet einen gewissen Aufwand, da für jeden APT die Aneignung umfassenden Domänenwissens erforderlich ist, soll es im Client berücksichtigt werden.

Zweitens sind die Erkennungsregeln konzentriert auf die Domäne von Angriffen, die auf der Netzwerkebene wahrnehmbar sind. So kann bspw. das Exfiltrieren von Daten auf den angegriffenen Geräten durch einen APT in Richtung eines externen Servers erkannt werden, nicht jedoch bspw. Änderungen in der PC-Konfiguration, die sich in Systemlogs niederschlagen können (z.B. Änderungen in der Registry-Datenbank auf Windows-Systemen). Entsprechende Regeln, die diese Angriffe identifizieren könnten, bestehen noch nicht.

Mit der Konzentrierung auf die Netzwerkebene einher geht, dass der Prototyp bisher nur mit Suricata-Logs verwendet wurde, die auf PCAP-Mitschnitten aufbauen. Da das YARA-Tool, in dessen Syntax die Erkennungsregeln verfasst sind, einen String-Matching-Ansatz verfolgt, kann grundsätzlich ein beliebiges Textlogformat hochgeladen und mittels YARA-Regeln ausgewertet werden. Je nach Logformat kann es jedoch sein, dass z.B. zusätzlich Informationen geliefert werden, die in anderen Logformaten nicht vorhanden sind, aber für die Evaluierung bzw. Bekämpfung des Angriffs durch den User nützlich sein könnten. Denkbar wären hier etwa Angaben zum Schweregrad einer Angriffsaktivität.

Derartige Informationen zu extrahieren, könnte für den User unter Umständen nützlich sein, würde jedoch unter Umständen Anpassungen im Quellcode mit sich ziehen, um diese in der Auswertung zu berücksichtigen oder darzustellen. Bereits in seiner aktuellen Form ist der Prototyp jedoch in der Lage, Angriffsindikatoren beliebiger Logformate zu erkennen, gegeben die entsprechenden YARA-Regeln, die Logdateien auf das Vorhandensein dieser Indikatoren prüfen.

8.1.3. Auswertungsalgorithmus

Der Auswertungsalgorithmus erhält als Input die durch YARA-Erkennungsregeln identifizierten Matches für Angriffsindikatoren und extrahiert aus ihnen eine Auswertung, die dem User mitteilt, in welchem Ausmaß für welchen APT Angriffsaktivitäten in den hochgeladenen Logdateien festgestellt wurde. Ein Hauptziel ist folglich, dem User eine fundierte Einschätzung zu liefern, welcher APT verantwortlich für den Angriff sein könnte, und ihn somit zur Angriffsbekämpfung zu befähigen.

Auf Basis des Scope der vorliegenden Arbeit fokussiert sich der gegenwärtige Algorithmus auf Aussagen zur Anzahl erkannter Techniken je APT sowie zur Anzahl über alle analysierten APTs einzigartiger Techniken je APT. Damit

werden bspw. weder der Schweregrad eines konkreten Angriffs in die Betrachtung einbezogen noch die Aussagekraft eines Indikators für das Vorhandensein eines spezifischen APT. Davon wurde abgesehen, da angesichts begrenzter Datenlage zu Angriffsindikatoren sowie begrenztem Testmaterial für die einzelnen APTs eine solche Genauigkeit in der Gewichtung nicht gewährleistet werden kann – siehe hierzu auch Abschnitt 7 und insbesondere Abschnitt 7.3.

Gewichtungen dieser Art können vor allem bei verbesserter, umfassenderer Datenlage bezüglich der Anzahl von Erkennungsregeln umgesetzt werden. Eine Verfeinerung des Erkennungsalgorithmus wäre z.B. auch mit Blick auf die Domäne der Aktivität denkbar (welche Aktivität etwa auf Netzwerkebene, welche lokal stattfindet).

8.1.4. Fallevaluierung durch User

Die im Rahmen der Masterarbeit durchgeführte Evaluierung ist im Wesentlichen auf die Überprüfung der Funktionalität beschränkt. Es bietet sich daher an, in einem weiteren Schritt auch die Usablity und den Funktionsumfang der Fallevaluierung zu erweitern. Da diese zusätzlichen Evaluierungsschritte jedoch für den Aspekt der korrekten Funktionalität des Systems nicht von zentraler Bedeutung sind, wurden sie auch in dieser Masterarbeit nur sehr rudimentär berücksichtigt. Um eine einsatzfähige Version des Systems zu produzieren, wären daher noch zusätzliche Schritte möglich:

Derzeit werden etwa die in den hochgeladenen Logdateien identifizierten Angriffstechniken im Client-Interface auf Fallebene aufgelistet. Denkbar wäre eine Erweiterung dieser Darstellung um eine Zuordnung der Techniken zu den jeweiligen APTs, für welche diese identifiziert wurden. Aktuell wird nur die Gesamtzahl entdeckter Techniken pro APT sowie die Anzahl jener Techniken pro APT, die nur für dieses erkannt wurden, angezeigt.

In der Falldokumentation in Form eines JSON, das zu einem Fall heruntergeladen werden kann, ist diese Aufzählung der Techniken nach APTs bereits vorhanden, einschließlich der Anzahl von Indikatoren, die für eine Technik angeschlagen haben. Diese Dokumentationsdatei, gedacht als eine Art „Expertensicht“, könnte ebenso verfeinert werden, z.B. indem die für eine Technik identifizierten Indikatoren explizit aufgelistet werden.

Ebenso könnten z.B. die von MITRE definierten Angriffsphasen dargestellt werden (in Textform oder auch grafisch aufbereitet). Jede Angriffstechnik ist durch MITRE grundsätzlich einer Angriffsphase bzw. einem Angriffszweck, genannt „tactic“, zugeordnet.[Theg] Eine Evaluierung, die diese Phasen explizit für den User darstellt, könnte das Verständnis eines Angriffs vertiefen.

Die etwa in den Abschnitten 8.1.2 sowie 8.1.3 beschriebenen Änderungen würden auch Anpassungen der Darstellung der Fallevaluierung nach sich ziehen. Man könnte bspw. die Darstellung um eine genauere Zuordnung der Logquellen erweitern. So würde z.B. ersichtlich werden, welcher Angriff auf der Netzwerkebene vorgenommen wurde und welcher lokal vorgenommen wurde. Auch eine

mögliche künftige Gewichtung nach Schweregrad einer Angriffstechnik könnte in der Falldarstellung abgebildet werden.

Diese Ergänzungen der Darstellung der Fallevaluierung können in der Client-GUI implementiert werden, in der JSON-Falldokumentation oder in beiden. Da die Genauigkeit der Darstellung auf Kosten der Übersichtlichkeit der Darstellung gehen kann und umgekehrt, würde sich in der Weiterentwicklung generell die Frage stellen, welche Informationen bereits in der GUI des Prototyps dargestellt werden und welche der umfassenden Expertensicht in Form der JSON-Falldokumentation vorbehalten sind. Dafür könnte etwa das Einholen von Userfeedback nützlich sein.

Aktuell wird im Client auf die MITRE ATT&CK-Datenbank zurückgegriffen, um dem Nutzer Vorschläge zur Angriffsbekämpfung bereitzustellen. Diese Vorschläge könnten erweitert werden um einen Verweis auf weitere Quellen oder Anlaufstellen, etwa staatliche CERTs. Dies kann insbesondere dann nützlich sein, wenn die Vorschläge zur Angriffsbekämpfung aus der MITRE-Datenbank ausgeschöpft sind, ohne den erhofften Erfolg gezeigt zu haben.

8.1.5. Deployment

Derzeit kann der Client-Prototyp ausschließlich lokal ausgeführt werden. Um ihn über das Internet bereitzustellen, sind noch einige Änderungen am Deployment erforderlich. Dies reicht vom Webhoster, auf dem der Client ausgeführt und über welchen er im Internet erreichbar ist, bis zu zusätzlichen Sicherheitsmaßnahmen. Diese wären erforderlich, da sich mit dem Zugänglichmachen der Anwendung über das Internet die Angriffsfläche erhöht. Ein unerlässlicher Schritt wäre, die Verbindung zwischen Client und User über SSL zu verschlüsseln. Im laufenden Betrieb müsste natürlich auch auf best practices wie das zeitnahe Aktualisieren des für SSL erforderlichen Zertifikats sowie das Einspielen von Sicherheitsupdates der einzelnen Komponenten (sowohl auf Ebene der Anwendung selbst als auch auf Ebene des Hosts) geachtet werden. In einem laufenden Online-Betrieb müsste auch die Auslastung der Server, die die Anwendung bereitstellen, beobachtet werden. Die im Prototyp verwendeten Tools wie etwa Nginx bieten Möglichkeiten, mit starker Auslastung sowie Nebenläufigkeiten in der Anwendung effizient umzugehen.[Matb]

8.2. Zentrale Schlussfolgerungen

Wie zu Beginn der Masterarbeit geschildert wurde, nehmen Cyberangriffe zu und bedrohen zunehmend die kritische Infrastruktur sowie ganze Wirtschaftszweige. APT Angriffe stellen dabei ganz besonders gefährliche und mächtige Cyber Angriffe dar. Um sich schützen zu können, wird die informationsunterstützende Analyse und Klassifizierung von Angriffen immer wichtiger. Dies betrifft sowohl die Vorbereitungsphase auf einen Angriff, als auch das Incident Handling. Hierbei

zählt in der Praxis die zeitgerechte Bereitstellung qualitätsgesicherter Information noch immer zu den wichtigsten Faktoren. Allerdings ist ein bedeutender Anteil an Organisationen auf Cyberangriffe unzureichend vorbereitet. IT Security Mitarbeiter sind mit einer Flut von Informationen konfrontiert und hierauf basierend Entscheidungen zu treffen, wird immer komplexer. Um möglichst sichere Entscheidungen treffen zu können, haben sich in der BWL Entscheidungsmodelle bewährt sowie der OODA Loop in der IT Security Domäne etabliert. Das Anliegen der vorliegenden Masterarbeit war es, Entscheidungsmodelle zu kombinieren und mit dem OODA Loop zu vereinen, um daraus ein für die IT Security Domäne relevantes Entscheidungsmodell abzuleiten. Der Schwerpunkt in der Masterarbeit liegt hierbei auf den zwei O-Phasen des OODA Loops (Observe und Orient), die der Detect Phase des NIST Frameworks zuzuordnen sind, um eine Entscheidung, durch Sammlung von Informationen und Orientierung anhand dieser Informationen, vorzubereiten. Dieses Entscheidungsmodell wurde im Anschluss als Prototyp implementiert, um bestimmte APT Angriffe anhand der MITRE ATT&CK Wissensdatenbank zu identifizieren und bewährte Gegenmaßnahmen vorzuschlagen.

Die Anwendung des in dieser Masterarbeit entworfenen Entscheidungsmodells bietet vielversprechende Erkenntnisse an, anhand derer die Beantwortung der in der Einleitung definierten zentralen Fragestellung (*„Wie können betriebswirtschaftliche Entscheidungsmodelle zur Unterstützung des Cyber Security Managements angewendet werden und bringen sie eine wesentliche Verbesserung zur Unterstützung des Incident Handling?“*) gelingen kann.

Die Kombination der vorgestellten Entscheidungsmodelle zu einem integrierten Entscheidungsmodell, das den OODA Loop komplettiert und dadurch ein High-Level SIEM als DSS für SOCs bereitstellt, bietet viel Potential, um das Cyber Security Management bzw. Indicent Handling zu unterstützen, denn insgesamt vermag der Client-Prototyp verlässlich solche Angriffe zu erkennen, die spezifisch einem der berücksichtigten APTs entsprechen. Der Prototyp ist auch in der Lage, Indikatoren über den gesamten „Life Cycle“ (entsprechend der Mitre Phasen) eines Angriffs zu finden, jedenfalls soweit dieser auf Netzwerkebene abbildbar ist. Angesichts der Komplexität des Vorgehens von APTs und der begrenzten Ressourcen lieferte die Evaluierung damit insgesamt ein zufriedenstellendes Resultat. Dadurch kann der Prototyp nicht nur in der Detect Phase des NIST Frameworks von Vorteil sein, sondern auch in der Respond Phase, wenn Entscheidungen benötigt werden, um Gegenmaßnahmen einzuleiten. Nachdem der Prototyp die Observe/Orient Phase des OODA Loops gut unterstützt, kann er ein SOC systematisch unterstützen.

Auch das integrierte Entscheidungsmodell für sich bietet eine wesentliche Verbesserung auf Basis seines Double Loop Learning Konzepts: Denn der innere Loop des Entscheidugsmodells bietet einer Organisation die Möglichkeit zu

hinterfragen, ob man alle seine sensorischen Möglichkeiten ausgeschöpft hat, um einen Angriff identifizieren zu können, sprich sich selber die Frage zu stellen: „Are we doing things right?“ Darüber hinaus bietet der äußere Loop eine gewisse generelle Reflexionsfähigkeit einer Organisation an, indem er hilft zu hinterfragen, ob die vorhanden Sensoren bzw. organisatorischen Abläufe überhaupt ausreichen können, um einen Angriff ausreichend zu identifizieren, sprich sich selber die Frage zu stellen: „Are we doing the right things?“ Dies soll einem zentralen Problem von DSS entgegenwirken. Denn ein DSS wird „schlechte“ Entscheidungen oder schlecht gezogene Schlussfolgerungen, die sich aus dem Stellen der „falschen“ Fragen ergeben, nicht beseitigen. Die Ergebnisse eines DSS müssen generell kritisch geprüft und zusammen mit dem bestehenden Verständnis der breiteren Geschäfts- oder Anwendungsbereiche verwendet werden.[FE09, S.31]

Der Prototyp könnte in der Praxis auch bei Simulationen eingesetzt werden. Es könnten z.B. Logfiles von bekannten Hacks hochgeladen werden und in einer Simulation die Unterstützung des Prototyps genutzt werden, um einen Angriff zu erkennen bzw. die Gegenmaßnahmen durchzuspielen, damit eine Organisation lernen kann, wie sie sich bei einem Angriff verhält bzw. bei der Abwehr verbessert.

APT Angriffe im Speziellen bzw. Cyberattacken im Allgemein sind ein Problem, das bleiben wird. In unserer heutigen Zeit, in der ein verstärkter Einsatz von künstlicher Intelligenz in unserem Berufs- und Privatleben zu verzeichnen ist, bleibt es spannend, zu beobachten, wie sich dieser Trend in Cyberattacken manifestieren wird. Das vorliegende Entscheidungsmodell kann aufgrund seiner kompakten und doch präzisen Gestaltung wertvolle erste Anhaltspunkte und eine Orientierung für das Handling von APT Cyberattacken liefern und ist daher ein Beitrag zur Unterstützung des Incident Handling. Bei entsprechendem vollen Ausbau des Systems kann der in der Masterarbeit entwickelte Ansatz daher ein sehr wesentlicher Bestandteil des Arsenals in der Abwehr von APT Angriffen werden.

A. Ergänzte Einträge in Logs generierter PCAPs für Tests 1–3

A.1. Test 1: Turla

Ergänzt in *fast_Misc_1*:

```
 11/05/2022-15:18:36.471351  [**] [1:2016277:5] ET
EXPLOIT_KIT Metasploit CVE-2012-1723 Path (Seen in Unknown EK
) 10/29/12 [**] [Classification: Exploit Kit Activity
Detected] [Priority: 1] {TCP} 192.168.0.103:80  ->
10.0.0.81:49951
```

Ergänzt in *fast_Misc_2*:

```
 11/06/2022-14:56:40.345571  [**] [1:2012084:3] ET NETBIOS
Microsoft Windows SMB Client Race Condition Remote Code
Execution [**] [Classification: Attempted User Privilege Gain
] [Priority: 1] {TCP} 192.168.0.104:80 -> 10.0.0.73:50636
```

Ergänzt in *fast_Misc_3*:

```
 11/06/2022-15:27:20.345321  [**] [1:2010781:3] ET POLICY
PsExec service created [**] [Classification: A suspicious
filename was detected] [Priority: 2] {TCP} 192.168.0.109:445
-> 10.0.0.73:445
```

Ergänzt in *fast_Misc_4*:

```
 11/12/2022-11:18:21.539019  [**] [1:2030236:1] ET MALWARE
TURLA NETFLASH CnC [**] [Classification: Malware Command and
Control Activity Detected] [Priority: 1] {TCP}
10.0.0.157:50274  -> 192.168.0.109:80
```

Ergänzt in *fast_Misc_5*:

```
 11/12/2022-12:07:40.684912  [**] [1:2030702:1] ET POLICY
[401TRG] DropBox Access via API (SNI) [**] [Classification:
Potential Corporate Privacy Violation] [Priority: 1] {TCP}
10.0.0.157:51636 -> 192.168.0.109:80
```

A.2. Test 2: APT19

Ergänzt in *fast_Misc_1*:

```
 11/05/2022-15:20:02.255085  [**] [1:2025085:3] ET WEB_CLIENT
 Hostile Microsoft Rich Text File (RTF) with corrupted
listoverride [**] [Classification: Attempted User Privilege
Gain] [Priority: 1] {TCP} 192.168.0.103:80 -> 10.0.0.81:49951
```

Ergänzt in *fast_Misc_2*:

```
 11/06/2022-14:57:48.430815  [**] [1:2024550:4] ET MALWARE
Likely Malicious Windows SCT Download MSXMLHTTP M1 [**] [
Classification: A Network Trojan was detected] [Priority: 1]
{TCP} 10.0.0.73:50636 -> 192.168.0.104:80
```

Ergänzt in *fast_Misc_3*:

```
 11/06/2022-15:35:49.592712  [**] [1:2029743:2] ET MALWARE
Cobalt Strike Malleable C2 (OneDrive) [**] [Classification:
Malware Command and Control Activity Detected] [Priority: 1]
{TCP} 10.0.0.73:51734 -> 192.168.0.109:80
```

Ergänzt in *fast_Misc_4*:

```
 11/12/2022-11:20:06.334030  [**] [1:2027211:3] ET MALWARE
Outbound POST Request with Base64 ps PowerShell Command
Output M1 [**] [Classification: A Network Trojan was detected
] [Priority: 1] {TCP}  10.0.0.157:50274 -> 192.168.0.109:80
```

Ergänzt in *fast_Misc_5*:

```
 11/12/2022-12:18:32.543512  [**] [1:2027700:5] ET ADWARE_PUP
 Fun Web Products Spyware User-Agent (FunWebProducts) [**] [
Classification: Possibly Unwanted Program Detected] [Priority
: 2] {TCP} 10.0.0.157:64414 -> 192.168.0.109:80
```

A.3. Test 3: APT28

Ergänzt in *fast_Misc_1*:

```
 11/05/2022-15:21:15.089831  [**] [1:2024364:4] ET SCAN
Possible Nmap User-Agent Observed [**] [Classification: Web
Application Attack] [Priority: 1] {TCP} 10.0.0.81:49951 ->
192.168.0.109:80
```

sowie

```
 11/05/2022-15:22:05.089831  [**] [1:2010215:6] ET SCAN SQL
Injection Attempt (Agent uil2pn) [**] [Classification: Web
Application Attack] [Priority: 1] {TCP} 192.168.0.103:80 ->
10.0.0.81:59420
```

Ergänzt in *fast_Misc_2*:

```
 11/06/2022-14:59:03.961203  [**] [1:2026757:4] ET MALWARE
Observed Malicious SSL Cert (SedUploader) [**] [
Classification: A Network Trojan was detected] [Priority: 1]
{TCP} 192.168.0.104:80 -> 10.0.0.73:50632
```

Ergänzt in *fast_Misc_3*:

```
 11/06/2022-15:47:11.042897  [**] [1:2220017:1] SURICATA SMTP
 Mime boundary length exceeded [**] [Classification: Generic
Protocol Command Decode] [Priority: 3] {TCP} 10.0.0.73:52609
-> 192.168.0.109:80
```

Ergänzt in *fast_Misc_4*:

```
 11/12/2022-11:22:35.058371  [**] [1:2017206:2] ET INFO
Obfuscated Eval String 1 [**] [Classification: A Network
Trojan was detected] [Priority: 1] {TCP} 192.168.0.109:80 ->
10.0.0.157:50274
```

sowie

```
 11/12/2022-11:23:49.540834  [**] [1:2031628:2] ET MALWARE
Suspected Fancy Bear (APT28) Maldoc CnC [**] [Classification:
 Malware Command and Control Activity Detected] [Priority: 1]
 {TCP} 10.0.0.157:50274 -> 192.168.0.255:80
```

Ergänzt in *fast_Misc_5*:

```
 11/12/2022-12:33:49.089831  [**] [1:2019584:3] ET MALWARE
CORESHELL Malware Response from server [**] [Classification:
A Network Trojan was detected] [Priority: 1] {TCP}
192.168.0.109:80 -> 10.0.0.157:65197
```

sowie

```
 11/12/2022-12:35:31.664231  [**] [1:2017769:2] ET HUNTING
SUSPICIOUS Java Request With Uncompressed JAR/Class Hex
Encoded Class file [**] [Classification: Potentially Bad
Traffic] [Priority: 2] {TCP} 192.168.0.109:80 ->
10.0.0.157:65197
```

Literaturverzeichnis

[And] Andreas Danzer. Jedes dritte Unternehmen erkennt Cybervorfälle im eigenen System nicht. https://www.derstandard.at/story/2000144500213/jedes-dritte-unternehmen-erkennt-cyber-vorfaelle-im-eigenen-system-nicht. zugegriffen: 2023-04-29.

[Bev13] Jason Bevis. *Creating and Maintaining a SOC - The details behind successful Security Operations Centers*. McAfee Inc., 2013.

[Boy18] John Boyd. *A Discourse on Winning and Losing*. Air University Press, 2018.

[Bun] Bundesamt für Sicherheit in der Informationstechnik. Advanced Persistent Threat. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Gefaehrdungen/APT/apt_node.html. zugegriffen: 2023-06-10.

[CG11] E. Cayirci and Reyhaneh Ghergherehchi. Modeling cyber attacks and their effects on decision process. 12 2011.

[CGL20] Markus Christen, Bert Gordijn, and Michele Loi. *The Ethics of Cybersecurity*. 01 2020.

[Cha] Charles Griffiths. The Latest 2023 Cyber Crime Statistics (updated June 2023). https://aag-it.com/the-latest-cyber-crime-statistics/#:~:text=Headline%20Cyber%20Crime%20Statistics&text=1%20in%202%20American%20internet,the%20first%20half%20of%202022. zugegriffen: 2023-06-10.

[Daf13] Richard Daft. *Organizational Theory & Design (11th ed.)*. South-Western, 2013.

[Diga] DigitalOcean, LLC. How To Set Up Flask with MongoDB and Docker. https://www.digitalocean.com/community/tutorials/how-to-set-up-flask-with-mongodb-and-docker. zugegriffen: 2023-03-11.

[Digb] DigitalOcean, LLC. How To Set Up uWSGI and Nginx to Serve Python Apps on Ubuntu 14.04. https://www.digitalocean.com/community/tutorials/how-to-set-up-uwsgi-and-n

ginx-to-serve-python-apps-on-ubuntu-14-04. zugegriffen: 2023-02-05.

[Doca] Docker Inc. Compose file versions and upgrading. https://docs.docker.com/compose/compose-file/compose-versioning/. zugegriffen: 2023-03-12.

[Docb] Docker, Inc. Docker - Develop faster. Run anyware. https://www.docker.com/. zugegriffen: 2023-03-11.

[Docc] Docker, Inc. Docker Compose Overview. https://docs.docker.com/compose/. zugegriffen: 2023-03-11.

[Docd] Docker Inc. Install Docker Engine. https://docs.docker.com/engine/install/. zugegriffen: 2023-03-12.

[Doce] Docker, Inc. Use containers to Build, Share and Run your applications. https://www.docker.com/resources/what-container/. zugegriffen: 2023-03-11.

[FE09] R Faiz and Eran Edirisinghe. Decision making for predictive maintenance in asset information management. *Interdisciplinary Journal of Information, Knowledge, and Management*, 4, 01 2009.

[fSidI17] Bundesamt für Sicherheit in der Informationstechnik. Bsi-standard 200-1: Managementsysteme für informationssicherheit (isms). 2017.

[Fur21] Steven Furnell. The cybersecurity workforce and skills. *Computers Security*, 100:102080, 2021.

[Gab] Gabler Wirtschaftslexikon - Robert Gillenkirch. Entscheidungstheorie. https://wirtschaftslexikon.gabler.de/definition/entscheidungstheorie-32315/version-255858. zugegriffen: 2023-04-29.

[Gita] GitHub, Inc. The ATTCK spec - Extensions of the STIX spec. https://github.com/mitre-attack/attack-stix-data/blob/master/USAGE.md#extensions-of-the-stix-spec. zugegriffen: 2023-03-05.

[Gitb] GitHub, Inc. yara-python. https://github.com/VirusTotal/yara-python. zugegriffen: 2023-03-05.

[GP14] Ibrahim Ghafir and Vaclav Prenosil. Advanced persistent threat attack detection: An overview. *International Journal Of Advances In Computer Networks And Its Security*, 12 2014.

[HAC+16] Diane Henshel, Alexander Alexeev, Mariana Cains, Jeff Rowe, Hasan Cam, Blaine Hoffman, and Iulian Neamtiu. Modeling cybersecurity risks: Proof of concept of a holistic approach for integrated risk quantification. In *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pages 1–5, 2016.

[Ine] Ines Panker. The Layered World Of Web Development: Why I Need NGINX And UWSGI To Run A Python App? `http://www.ines-panker.com/2020/02/16/nginx-uwsqi.html`. zugegriffen: 2023-03-11.

[Jet] Jet Brains s.r.o. PyCharm: The Python IDE For Professional Developers. `https://www.jetbrains.com/pycharm/`. zugegriffen: 2023-03-05.

[Jon13] Gareth Jones. *Organizational Theory, Design, and Change (7th ed.)*. Pearson, 2013.

[JSM19] Mohammad S. Jalali, Michael Siegel, and Stuart Madnick. Decision-making and biases in cybersecurity capability development: Evidence from a simulation game experiment. *The Journal of Strategic Information Systems*, 28(1):66–82, 2019.

[Ken] Kenneth Reitz Real Python. The Hitchhiker's Guide to Python - Web Applications Frameworks. `https://docs.python-guide.org/scenarios/web/`. zugegriffen: 2023-02-05.

[Kin] Kinsta, Inc. How To Set Up a Reverse Proxy (Step-By-Steps for Nginx and Apache). `https://kinsta.com/blog/reverse-proxy/`. zugegriffen: 2023-03-11.

[KK22] Carson Zimmerman Kathryn Knerler, Ingrid Parker. *11 Strategies of a World-Class Cybersecurity Operations Center*. The MITRE Corporation, 2022.

[Kle98] Gary Klein. *Sources of Power. How People Make Decisions (20th Anniversary Edition)*. The MIT Press, 1998.

[KPM] KPMG. The supply chain trends shaking up 2023. `https://kpmg.com/xx/en/home/insights/2022/12/the-supply-chain-trends-shaking-up-2023.html#:~:text=71%25%20of%20global%20companies%20highlight,spent%20managing%20low%2Ddemand%20items.` zugegriffen: 2023-06-10.

[KS82] Sara Kiesler and Lee Sproull. Managerial response to changing environments: Perspectives on problem sensing from social cognition. *Administrative Science Quarterly*, 27(4):548–570, Dezember 1982.

[KVBAO16] Oluyomi Kabiawu, Jean-Paul Van Belle, and Michael Adeyeye Oshin. Designing a knowledge resource to address bounded rationality and satisficing for ict decisions in small organizations. *The Electronic Journal of Information Systems in Developing Countries*, 73:1–18, März 2016.

[LLPE10] Hyun Lee, Byoungyong Lee, Kyungseo Park, and Ramez Elmasri. Fusion techniques for reliable information: A survey. *JDCTA*, 4:74–88, April 2010.

[LS14] Mark Lee and David Stinson. Organizational decision making models: Comparing and contrasting to the stinson wellness model. *European Journal of Management*, 14(3):13–28, Oktober 2014.

[Lyn11] Andrew Lynch. A novel operational decision making method for smes. 2011.

[Mata] Matt Adams. Secure MongoDB with Docker. `https://web.archive.org/web/20220524121258/https:/offhourscoding.com/secure-mongodb-with-docker/`. zugegriffen: 2023-03-11.

[Matb] Matt Williams. Scaling Web Applications with NGINX, Part 1: Load Balancing. `https://www.nginx.com/blog/scaling-web-applications-nginx-part-load-balancing/`. zugegriffen: 2023-05-18.

[Med] Medium. What is WSGI (Web Server Gateway Interface)? `https://medium.com/analytics-vidhya/what-is-wsgi-web-server-gateway-interface-ed2d290449e`. zugegriffen: 2023-03-11.

[MFM+19] Andrew M'manga, Shamal Faily, John McAlaney, Chris Williams, Youki Kadobayashi, and Daisuke Miyamoto. A normative decision making model for cyber security. *Information and Computer Security*, April 2019.

[Miga] Miguel Grinberg. Handling File Uploads With Flask. `https://blog.miguelgrinberg.com/post/handling-file-uploads-with-flask`. zugegriffen: 2023-03-25.

[Migb] Miguel Grinberg. The Flask Mega-Tutorial Part V: User Logins. `https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-v-user-logins/page/2`. zugegriffen: 2023-03-25.

[Mik] Mike Huls. Keep your code secure by using environment variables and env files. `https://towardsdatascience.com/keep-your-code-secure-by-using-environment-variables-and-env-files-4688a70ea286`. zugegriffen: 2023-03-18.

[Mit] Mitre. MITRE ATT&CK. https://attack.mitre.org. zugegriffen: 2023-06-10.

[Mona] MongoDB Developer Community. Escaping a string before inserting/updating with pymongo. https://www.mongodb.com/community/forums/t/escaping-a-string-before-inserting-updating-with-pymongo/720/4. zugegriffen: 2022-03-25.

[Monb] MongoDB, Inc. Compass. The GUI for MongoDB. https://www.mongodb.com/products/compass. zugegriffen: 2023-02-19.

[Monc] MongoDB, Inc. PyMongo. https://pymongo.readthedocs.io/en/stable/. zugegriffen: 2023-03-05.

[Mond] MongoEngine Authors Revision ac8ba504. MongoEngine User Documentation. https://mongoengine-odm.readthedocs.io. zugegriffen: 2023-03-05.

[MRT76] Henry Mintzberg, Duru Raisinghani, and André Théorêt. The structure of ünstructured"decision processes. *Administrative Science Quarterly*, 21(2):246–275, 1976.

[Nik] Nikita Kakuev. Understanding front-end data visualization tools ecosystem in 2021. https://cube.dev/blog/dataviz-ecosystem-2021. zugegriffen: 2023-05-18.

[NSR00] Emma Norling, Liz Sonenberg, and Ralph Rönnquist. Enhancing multi-agent based simulation with human-like decision making strategies. pages 214–228, 01 2000.

[Num] NumPy. NumPy. https://numpy.org. zugegriffen: 2023-03-05.

[OAS] OASIS Cyber Threat Intelligence. Stix - Introduction to STIX. https://oasis-open.github.io/cti-documentation/stix/intro.html. zugegriffen: 2023-02-18.

[ORF] ORF.at, Agenturen. Italien im Visier russischer Hacker. https://orf.at//stories/3309793/. zugegriffen: 2023-04-29.

[oST18] National Institute of Standards and Technology. *Framework for Improving Critical Infrastructure Cybersecurity - Version 1.1*. National Institute of Standards and Technology, 2018.

[OWAa] OWASP. A03 injection - owasp top 10:2021. https://owasp.org/Top10/A03_2021-Injection/. zugegriffen: 2022-03-25.

[OWAb] OWASP. Path traversal. https://owasp.org/www-community/attacks/Path_Traversal. zugegriffen: 2022-03-25.

[Pala] Pallets. Flask: web development, one drop at a time. `https://flask.palletsprojects.com/en/2.2.x/`. zugegriffen: 2023-03-05.

[Palb] Pallets. Jinja. `https://jinja.palletsprojects.com/en/3.1.x/`. zugegriffen: 2023-03-05.

[Palc] Pallets. MongoDB with MongoEngine. `https://flask.palletsprojects.com/en/2.2.x/patterns/mongoengine/`. zugegriffen: 2023-03-05.

[Pald] Pallets. Templates - jinja setup. `https://flask.palletsprojects.com/en/2.2.x/templating/#jinja-setup`. zugegriffen: 2022-03-25.

[Pale] Pallets. Werkzeug utilities - general helpers. `https://werkzeug.palletsprojects.com/en/2.2.x/utils/#werkzeug.utils.secure_filename`. zugegriffen: 2022-03-25.

[Pin86] Lawrence T. Pinfield. A field evaluation of perspectives on organizational decision making. *Administrative Science Quarterly*, 31(3):365–388, 1986.

[Pro] Project Management Docs. Use case document template. `https://www.projectmanagementdocs.com/template/project-documents/use-case-document/`. zugegriffen: 2022-12-10.

[Qui18] Gerald Quirchmayr. Cert komm ii kiras-projekt: Endbericht. Project report, Austria, 2018.

[Reaa] ReadTheDocs. flask-login. `https://flask-login.readthedocs.io/en/latest/`. zugegriffen: 2023-03-05.

[Reab] ReadTheDocs. Flask-Markdown. `https://flask-markdown.readthedocs.io/en/master/`. zugegriffen: 2023-03-05.

[Reac] ReadTheDocs. STIX 2 Python API Documentation. `https://stix2.readthedocs.io/en/latest/`. zugegriffen: 2023-03-05.

[Read] ReadTheDocs. The uWSGI project. `https://uwsgi-docs.readthedocs.io/en/latest/`. zugegriffen: 2023-03-05.

[RGSM18] Maria Rashidi, Maryam Ghodrat, Bijan Samali, and Masoud Mohammadi. Decision support systems. In Maria Pomffyova, editor, *Management of Information Systems*, chapter 2. IntechOpen, Rijeka, 2018.

[Riz14] Nina Rizun. Methodology of intellectual express evaluation of the decision-making effectiveness. *Economics and Business Communication Challenges*, pages 209–229, Dezember 2014.

[Ros14] Phil Rosenzweig. The benefits—and limits—of decision models. *McKinsey Quarterly*, Februar 2014.

[Ser] Server Fault. Why do I need nginx when I have uWSGI. `https://serverfault.com/questions/590819/why-do-i-need-nginx-when-i-have-uwsgi/590833#590833`. zugegriffen: 2023-03-11.

[Sim60] Herbert Simon. *The New Science of Management Decision*. Prentice-Hall, 1960.

[Sim09] Herbert Simon. *Economics, bounded rationality, and the cognitive revolution*. Edward Elgar Publishing, 2009.

[Ski] Skill Suits. Core uWSGI – An Application Server. `https://skillsuites.com/core-uwsgi-an-application-server/`. zugegriffen: 2023-03-11.

[SLW02] Diane Strong, Yang Lee, and Richard Wang. Data quality in context. *Communications of the ACM*, 40, 08 2002.

[SPP85] William Stevenson, Jone Pearce, and Lyman Porter. The concept of "coalitionïn organization theory and research. *The Academy of Management Review*, 10(2):256–268, April 1985.

[SR93] Paul Schoemaker and J. Russo. A pyramid of decision approaches. *California Management Review*, 36, 10 1993.

[sta] statista - Taylor Petroc. Global market share held by operating systems for desktop PCs, from January 2013 to January 2023. `https://www.statista.com/statistics/218089/global-market-share-of-windows-7/`. zugegriffen: 2023-05-20.

[Teca] TechMormo. The Bridge Network Driver | Networking in Docker 6. `https://techmormo.com/posts/docker-networking-6-bridge-driver/`. zugegriffen: 2023-03-11.

[Tecb] TechTarget. decision support system (DSS). `https://www.techtarget.com/searchcio/definition/decision-support-system`. zugegriffen: 2023-04-29.

[Thea] The Matplotlib development team. Matplotlib: Visualization with Python. `https://matplotlib.org`. zugegriffen: 2023-03-05.

[Theb] The MITRE Corporation. APT 19. https://attack.mitre.org/groups/G0073/. zugegriffen: 2023-02-11.

[Thec] The MITRE Corporation. APT 28. https://attack.mitre.org/groups/G0007/. zugegriffen: 2023-02-11.

[Thed] The MITRE Corporation. ATT&CK® Navigator: APT19 (G0073). https://mitre-attack.github.io/attack-navigator//#layerURL=https%3A%2F%2Fattack.mitre.org%2Fgroups%2FG0073%2FG0073-enterprise-layer.json. zugegriffen: 2022-12-04.

[Thee] The MITRE Corporation. ATT&CK® Navigator: APT28 (G0007). https://mitre-attack.github.io/attack-navigator//#layerURL=https%3A%2F%2Fattack.mitre.org%2Fgroups%2FG0007%2FG0007-enterprise-layer.json. zugegriffen: 2022-12-04.

[Thef] The MITRE Corporation. ATT&CK® Navigator: Turla (G0010). https://mitre-attack.github.io/attack-navigator//#layerURL=https%3A%2F%2Fattack.mitre.org%2Fgroups%2FG0010%2FG0010-enterprise-layer.json. zugegriffen: 2022-12-04.

[Theg] The MITRE Corporation. Enterprise tactics. https://attack.mitre.org/tactics/enterprise/. zugegriffen: 2023-05-18.

[Theh] The MITRE Corporation. Mitre ATT&CK Update v9. https://attack.mitre.org/resources/updates/updates-april-2021/index.html. zugegriffen: 2023-02-04.

[Thei] The MITRE Corporation. Turla. https://attack.mitre.org/groups/G0010/. zugegriffen: 2023-02-11.

[Thej] The Open Information Security Foundation (OISF). Suricata. https://suricata.io/. zugegriffen: 2022-12-04.

[Wik] Wikipedia. uWSGI. https://de.wikipedia.org/wiki/UWSGI. zugegriffen: 2023-03-05.

[Wir] Wireshark. Wireshark. https://www.wireshark.org/. zugegriffen: 2022-12-04.

[WTFa] WTForms. Flask WTF. https://flask-wtf.readthedocs.io/en/1.0.x/. zugegriffen: 2023-03-05.

[WTFb] WTForms. WTForms. https://wtforms.readthedocs.io/en/3.0.x/. zugegriffen: 2023-03-05.

[Yar] Yara. Yara - The pattern matching swiss knife for malware researchers (and everyone else). https://virustotal.github.io/yara/. zugegriffen: 2023-02-12.

[ZZ17] Robert Zager and John Zager. Ooda loops in cyberspace: A new cyber-defense model. *Small Wars Journal*, Oktober 2017.