Information Security

Taktisches SIEM als Basis für ein SOC

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

eingereicht von

Dominik Handl, BSc

1710619807

im Rahmen des

Studienganges Information Security an der Fachhochschule St. Pölten

Betreuung

Betreuer/in: Dipl.-Ing. Gabor Österreicher, BSc

St. Pölten, 30. Juni 2021

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- diese Arbeit mit der vom Begutachter/von der Begutachterin beurteilten Arbeit übereinstimmt.

Der Studierende/Absolvent räumt der FH St. Pölten das Recht ein, die Diplomarbeit für Lehre- und Forschungstätigkeiten zu verwenden und damit zu werben (z.B. bei der Projektevernissage, in Publikationen, auf der Homepage), wobei der Absolvent als Urheber zu nennen ist. Jegliche kommerzielle Verwertung/ Nutzung bedarf einer weiteren Vereinbarung zwischen dem Studierenden/Absolventen und der FH St. Pölten.

Ort, Datum *Unterschrift*

Kurzfassung

Aufgrund der immer steigenden Komplexität von modernen IT-Systemen und des Anstiegs von erfolgreichen Angriffen auf diese Systeme, ist es notwendig, die IT-Sicherheitsüberwachung zu optimieren. Durch fehlende Mitarbeiterressourcen und den großen Umfang an Informationsquellen können einzelne Unternehmen oftmals keine detaillierte und andauernde Überprüfung der sicherheitsrelevanten Informationen durchführen. Aus diesem Grund werden immer häufiger diese Tätigkeiten an externe Unternehmen ausgelagert. Damit ein Betreiber diese Dienste effizient erbringen kann, ist es notwendig, ein skalierbares und teilautomatisiertes System zur Verfügung zu haben. Da sich Angriffsmuster und mögliche Schadsoftware regelmäßig anpassen, ist es erforderlich, aktuelle Informationen in diesem System zu pflegen. Diese Arbeit beschäftigt sich mit der Thematik, welche Anforderungen von einem solchen System mindestens erbracht werden müssen und mithilfe welcher Technologien diese umgesetzt werden können.

Im ersten Abschnitt werden die Anforderungen an das System erläutert und theoretische Grundlagen über die Implementierungsvarianten gesammelt. Hierbei wird darauf geachtet, dass das System aktuelle Informationen über neue Angriffsvarianten automatisiert integriert und somit die Mitarbeiterressourcen für den Betrieb dieser Systeme reduziert. Trotz der reduzierten Zeitressourcen soll dennoch die Erkennungsrate und die Qualität der Alarme erhöht werden. Damit diese Anforderungen umgesetzt werden können, werden unterschiedliche Quellen für Open-Source-Threat-Intelligence betrachtet. In den weiteren Abschnitten werden die gesammelten theoretischen Informationen in ein System implementiert. Bei der Implementierung wird darauf geachtet, dass das System skalierbar ist und schnell auf neue Bedürfnisse angepasst werden kann.

Das Ziel dieser Arbeit ist es, ein zentrales System für die Sammlung und Analyse von Eventdaten zur Verfügung zu stellen, die sicherheitsrelevant sind. Des Weiteren werden Integrationsmöglichkeiten von Open-Source-Threat-Intelligence-Systemen umgesetzt und die notwendigen Eventquellen für eine Erkennung von Angriffen implementiert. Das in dieser Arbeit entworfene System soll als Basis für einen Security-Operations-Center-(SOC-)Betreiber dienen und die grundlegenden Funktionen zur Verfügung stellen.

Abstract

Due to the ever-increasing complexity of modern IT systems and the rapid increase in successful attacks on these systems, it is necessary to optimize IT security-related monitoring. Due to a lack of staff resources and the large volume of information, individual companies are often unable to carry out detailed and continuous checks of security-relevant information. For this reason, these activities are increasingly outsourced to external companies. In order for an operator to provide these services efficiently, it is necessary to have a scalable and partially automated system available. Since attack patterns and possible malware adapt regularly, it is essential to maintain up-to-date information in this system. This thesis deals with the minimum requirements of such a system and which technologies can be used to implement them.

In the first section the requirements for the system are explained and theoretical information about the implementation variants is collected. In this context, it is important that the system integrates current information about new attack variants in an automated manner, thus reducing the staff resources needed to operate the systems. Despite the reduced time resources, the detection rate and the quality of the alarms should still be increased. In order to implement these requirements, different sources of open source threat intelligence are considered. In the further sections, the collected theoretical information is implemented into a system. During the implementation, it is important to ensure that the system is scalable and can be quickly adapted to new needs.

The goal of this thesis is to provide a central system for the collection and analysis of IT security relevant event data. Furthermore, integration possibilities of open source threat intelligence systems are implemented and the necessary event sources for a detection of attacks are implemented. The system designed in this thesis shall serve as a basis for a Security Operations Center operator and provide the basic functions.

Inhaltsverzeichnis

1. Einleitung

Aufgrund der immer steigenden Anzahl von erfolgreichen Angriffen auf IT-Systeme und die dadurch entstehenden Kosten für Betroffene, besteht die Notwendigkeit einer zeitnahen Erkennung eines bereits durchgeführten oder andauernden Angriffs. Im Jahr 2020 betrug die durchschnittliche Dauer, bis ein/e Angreifer/in in einem kompromittierten System entdeckt wurde, 56 Tage [1, S. 11]. Dieser Zeitraum ermöglicht es der angreifenden Person sensible Daten aus Unternehmen zu stehlen und die Verfügbarkeit der IT-Systeme einzuschränken oder einen Totalausfall dieser auszulösen. Der Schaden eines erfolgreichen Angriffs kann durch eine frühzeitige Erkennung reduziert werden [2, S. 61].

Da gezielte Angriffe oft so ausgelegt werden, dass diese von signaturbasierenden Antivirus-Programmen oder einem Intrusion-Protection-System (IPS) nicht erkannt werden, müssen für die Erkennung weitere Informationen verwendet werden. Ein Ansatz für eine solche Erkennung ist die Auswertung der generierten Event-Protokolle der Systeme. Durch eine zentrale Aufbewahrung und Analyse der Daten können über alle Systeme hinweg Auswertungen durchgeführt werden [3, S. 91-98]. Diese Funktion wird von Security-Information-and-Event-Management-(SIEM-)Lösungen erfüllt. Damit diese ein gewünschtes Ergebnis liefern können, ist es notwendig, entsprechende Personalressourcen für die Weiterentwicklung und Anpassung der implementierten Regelwerke bereitzustellen. Da diese Ressourcen oft von kleinen bis mittelgroßen Unternehmen nicht zur Verfügung gestellt werden können, werden diese SIEMs häufig nach der Implementierung nicht mehr gewartet und dienen dadurch nur mehr für eine Datenspeicherung und Archivierung.

In dieser Diplomarbeit soll eine mögliche zentrale Implementierung einer Analyseplattform entworfen werden. Diese Umsetzung soll für die Verarbeitung von mehreren SIEM-Lösungen aus unterschiedlichen Unternehmen geeignet sein. Durch diese Funktion soll es ermöglicht werden, dass ein Betreiber der Plattform die Analyse als Service an kleine bis mittelgroße Unternehmen anbieten kann. Diese Variante bietet den Nutzern des Service eine Möglichkeit, den Mangel an fachspezifischen Personalressourcen über externe Dienstleistungen zu lösen. Durch die zentrale Integration und Auswertung kann sichergestellt werden, dass das entsprechende Regelwerk für alle eingebundenen Umgebungen aktuell ist. Da eine erfolgreiche Erkennung von der Aktualität der Informationen abhängig ist, werden die aufgezeichneten und übertragenen Event-Daten der Systeme anhand von Open-Source-Threat-Intelligence-Informationen überprüft. Dadurch soll der Aufwand für den Betreiber reduziert und die Qualität der Erkennungsrate verbessert werden [4].

1.1. Problemstellung

Die Qualität der Alarme von dezentralisierten SIEM-Implementierung mit statischem Regelwerk ist durch den Wartungsaufwand stark limitiert. Dadurch werden viele Falschmeldungen erzeugt, welche dazu führen, dass Alarme die Relevanz für die bearbeitenden Personen verlieren. Die Wartbarkeit von dezentralisierten SIEMs mit statischem Regelwerk bietet eine zusätzliche Fehlerquelle durch die dezentralisierte Konfiguration und die Skalierbarkeit für einen Security-Operation-Center-(SOC-)Betrieb ist nicht gegeben [5]. Ohne einer Integration von externen Cyber-Security-Quellen müssen Regeln für die gewünschten Alarmierungen manuell erstellt und in mehreren Umgebungen eingepflegt werden. Die Aktualität der verwendeten Alarmierungsregeln ist somit direkt von den Personalressourcen abhängig. Notwendige Anpassungen für aktuelle Angriffsmethoden müssen hierbei zuerst von Mitarbeiter/innen festgestellt und umgesetzt werden. Diese Abhängigkeit führt dazu, dass aktuelle Angriffsmethoden erst verzögert oder nicht erkannt werden [6].
Aufgrund der oben genannten Problematiken soll eine zentrale Implementierung entworfen werden, welche mithilfe von Open-Source-Technologien aufgebaut wird. Diese Installation soll die Basis für ein SOC bilden. Da diese Lösung für mehrere Kundenumgebungen verwendet und zentral betrieben werden soll, muss eine Mandantenfähigkeit gewährleistet sein. Damit die Qualität der Alarme erhöht und die Erkennung von aktuellen Bedrohungen verbessert wird, soll eine Open-Source-Threat-Intelligence-Platform evaluiert und in das System eingebunden werden. Anhand dieser Indikatoren soll ein entsprechendes Regelwerk entworfen werden. Zusätzlich zu diesem Regelwerk sollen veröffentlichte SIEM-Alarmierungsregeln im Sigma-Format integriert werden. Ein wesentlicher Faktor dieser Integration stellt die automatische Aktualisierung dieses Regelwerks dar. Weiterhin soll aber die Wartbarkeit der Regeln für einzelne Umgebungen gegeben sein [7].
Infolge der hohen Anzahl an verschiedenen Datenquellen für ein SIEM-System ist es notwendig, die Relevanz dieser Daten zu kontrollieren und zu ermitteln, welche Daten als Mindestanforderung für die Funktionsfähigkeit der integrierten Thread-Intelligence-Informationen benötigt werden. Die übermittelten Events der ermittelten Systeme müssen anhand deren Notwendigkeit vorgefiltert werden. Ansonsten besteht die Gefahr, dass das zentrale System durch irrelevante Einträge überlastet wird.

1.2. Forschungsfragen

Aus den oben angeführten Problemstellungen ergeben sich folgende Forschungsfragen, welche im Zuge dieser Diplomarbeit beantwortet werden:

- Welche Open-Source-Threat-Intelligence-Lösungen können für die Anreicherung von empfangenen Events verwendet und wie können diese integriert werden?
- Wie können Open-Source-SIEM-Regeln in das System integriert und ein automatisches Update sichergestellt werden?
- Welche Logquellen müssen mindestens eingebunden werden, damit eine Erkennung der Open-Threat-Intelligence-Indicator-of-Comprise bzw. die Verwendung der SIEM-Regeln möglich ist?
- Wie können die empfangenen Events anhand der MITRE-ATT&CK-Matrix automatisiert kategorisiert werden?

1.3. Methoden

Für die Beantwortung der Forschungsfragen werden theoretische und empirische Analysen angewandt.

1.3.1. Theoretische Analyse

Im ersten Schritt werden mithilfe von theoretischen Analysen die zur Verfügung stehenden Open-Source-Projekte bewertet. Im Zuge dieser Analyse sollen die vorhandenen Projekte analysiert und ein Plan für das Basis-System erstellt werden. Da eine Verwendung der Lösung als Service-as-a-Service geplant ist, müssen die verwendeten Projekte einer entsprechenden Open-Source-Lizenzierung unterliegen.
Damit eine Integration von Open-Source-Threat-Intelligence-Produkten umgesetzt werden kann, werden die angebotenen Lösungen ermittelt. Deren Funktionsumfang wird überprüft und auf die Tauglichkeit in diesem Umfeld kontrolliert. Dafür werden Anforderungen an eine solche Plattform definiert und bewertet. Für die automatische Zuweisung von Events anhand der MITRE-ATT&CK-Matrix werden im ersten Schritt die theoretischen Informationen zu der Matrix ermittelt und aufbereitet. Anhand dieser Informationen werden die Möglichkeiten für eine automatische Zuweisung ermittelt. Zusätzlich werden die weiteren Verwendungsmöglichkeiten der MITRE-ATT&CK-Informationen im Regelwerk kontrolliert.

1.3.2. Empirische Analyse

Für die Beantwortung der Forschungsfragen ist es notwendig, dass die theoretisch ermittelten Informationen praktisch getestet werden. Als Ausgangslage muss eine Testumgebung mit der geplanten Basis-Infrastruktur

aufgebaut werden. Damit eine Integration der Open-Source-Threat-Intelligence-Lösungen und eine Zuweisung anhand der MITRE-ATT&CK-Matrix möglich ist, sind Eigenentwicklungen notwendig, welche in die Basis Infrastruktur integriert werden müssen.
Für das Testverfahren zur Ermittlung der Funktionalität ist es erforderlich, das System mit Events aus tatsächlichen Vorfällen zu versorgen. Für diesen Schritt werden einerseits Testsysteme installiert, welche die Events in Echtzeit übermitteln. Zusätzlich werden Möglichkeiten evaluiert, bereits vorhandene Events, welche tatsächliche Vorfälle beinhalten, in das System einzuspielen.

1.4. Abgrenzung

Die vorliegende Arbeit versucht mithilfe von öffentlich verfügbaren Produkten und Informationen eine zentrale Plattform zu entwickeln, welche die Wartungsaufwände verringert und Erkennungsraten von möglichen Angriffen verbessert. Diese Plattform soll als Grundlage für ein neues SOC dienen. Die Arbeit fokussiert sich auf die Informationsbeschaffung der öffentlichen Informationen, die Funktionsweise der einzelnen Komponenten und deren Zusammenarbeit. Des weiteren wird bei den einzelnen Schritten darauf geachtet, dass die Umgebung skalierbar ist und bei zukünftigen Anforderungen erweitert werden kann. Es wird jedoch nicht darauf eingegangen, wie die notwendigen Informationen in einem Enterprise-Umfeld gesammelt und optimal übertragen werden. Für die Umsetzung wurden Test-Systeme verwendet, welche die notwendigen Daten generieren und in das System übertragen. Des weiteren werden mögliche Anforderungen aus Normen oder Zertifizierungen nicht betrachtet. Diese Arbeit konzentriert sich auf die zentralen Komponenten und setzt die dezentralen Bestandteile voraus.

2. Grundlagen

Im ersten Schritt gilt es, theoretische Grundlagen zu erläutern, um ein grundlegendes Verständnis über notwendigen Komponenten des Systems zu bekommen.

2.1. Begriffserklärung

Eine SIEM-Implementierung kann für verschiedene Anwendungszwecke und Aufgaben eingesetzt werden. Einerseits bestehen Richtlinien und Anforderungen aus möglichen Compliance-Bestimmungen für die Aufzeichnung und Speicherung von verschiedenen Protokolldaten. Ein wesentliches Aufgabengebiet eines SIEM-Systems ist die Erkennung von Cyber-Security-Vorfällen und die Alarmierung an die zuständigen Personen. Im Zuge dieser Arbeit werden die SIEM-Funktionen und Aufgaben auf zwei Varianten aufgeteilt, welche in weiterer Folge als Tactical-SIEM und Compliance-SIEM bezeichnet werden.

2.1.1. Tactical-SIEM

Das Tactical-SIEM soll für die Erkennung und Alarmierung von Cyber-Security-Vorfällen verwendet werden. Mithilfe von Threat-Intelligence-Informationen und entsprechenden Regelwerken werden die definierten Personen beim Auftreten von bekannten Indikatoren alarmiert. Da dieses System hauptsächlich für die Erkennung und Alarmierung verwendet werden soll, gibt es keine speziellen Anforderungen an die Aufbewahrungsdauer der einzelnen Log-Einträge. Sollte zu einem späteren Zeitpunkt die Analyse von älteren Log-Einträgen notwendig sein, können diese auf dem entsprechenden Compliance-SIEM ausgewertet werden.

2.1.2. Compliance-SIEM

Das Compliance-SIEM dient zur Speicherung der Protokolldaten und zur Erfüllung der Compliance-Richtlinien. Die unterschiedlichen Endgeräte wie Server, Netzwerkkomponenten oder auch Storagekomponenten senden deren Log-Einträge über unterschiedliche Protokolle an diese SIEM-Implementierung. Für die Übertragung der Log-Einträge werden unter anderem herstellerspezifische Protokolle bei Implementierungen mit Software-Agents auf den Servern verwendet. Bei Netzwerkkomponenten oder anderen Appliances werden

die Log-Einträge oftmals über das Protokoll Syslog an das SIEM weitergeleitet. Da das Compliance-SIEM weniger Logik beinhaltet und hauptsächlich für das Speichern und Weiterleiten von bestimmten Logs notwendig ist, liegt das Hauptaugenmerk der Arbeit bei der Funktionsweise des Tactical-SIEM.

2.2. Anforderungen an das System

Da das Tactical-SIEM für mehrere Kundenumgebungen, welche wiederum ein Compliance-SIEM enthalten, verwendet werden soll, muss das Basissystem für das Tactical-SIEM mindestens folgende Anforderungen abdecken.

Anreicherung

Die an das System weitergeleiteten Log-Einträge müssen mit weiteren Informationen angereichert werden können, damit eine Erkennung von möglichen Vorfällen ermöglicht wird. Zusätzlich soll eine Anreicherung von Informationen auch für die Vereinfachung von weiteren Analysen durchgeführt werden.

Skalierbarkeit

Die Architektur sollte so gewählt werden, dass eine Skalierbarkeit gegeben ist. Da die Anzahl der empfangenen Events direkt abhängig von der Anzahl und Größe der eingebundenen Umgebungen ist, können sich die notwendigen Ressourcen kurzfristig ändern. Wird eine neue Umgebung an das Tactical-SIEM angebunden, kann die Anzahl der empfangenen Events verdoppeln.

Mandantenfähigkeit

Da mehrere Umgebungen in das zentrale Tactical-SIEM eingebunden werden, muss dies Lösung eine Trennung bzw. Filterung der Daten je nach Umgebung ermöglichen. Da es für die Umgebungen auch unterschiedliche Administrator/inn/en gibt, muss der Zugriff für diese Personengruppen auf die notwendigen Daten limitiert werden können.

Lizenzkosten und Verwendungszwecke

Ein Hauptaugenmerk bei der Implementierung des Tactical-SIEMs soll darauf gerichtet werden, dass das System mit Open-Source-Lösungen implementiert wird. Es ist nicht gewünscht, dass für die Umsetzungen kommerzielle Lösungen verwendet werden. Des Weiteren muss die Open-Source-Lizenzierung der einzelnen Komponenten es erlauben, dass die implementierte Umgebung auch kommerziell als Service angeboten werden kann.

2.3. Aufbau der Basisarchitektur

Aufgrund der weitverbreiteten Open-Source-Integrationen und der Skalierbarkeit wurde der Fokus bei der Auswahl der Basistechnologien auf eine ELK-Stack-Implementierung gesetzt. Ein ELK-Stack besteht aus den Produkten Elasticsearch, Logstash und Kibana.

2.3.1. Elasticsearch

Elasticsearch bietet die Möglichkeit einer verteilten Speicherung der Log-Dateien und ermöglicht somit eine skalierbare Lösung. Durch die Verteilung der Daten kann der Speicherplatz wie auch die Lese- und Schreib-Performance erweitert bzw. optimiert werden. Die Funktionalität eines Elasticsearch-Systems ist von folgenden Kernkomponenten abhängig:

Node

Als Node wird ein einzelner Elasticsearch-Server bezeichnet, welcher die Daten speichert und im Elasticsearch-Cluster für die Index- und Suchfunktionen teilnimmt.

Cluster

Ein Elasticsearch-Cluster besteht aus mindestens einem Node. Die maximale Anzahl der Nodes ist nicht begrenzt. Die im Cluster gespeicherten Daten sind auf die teilnehmenden Nodes aufgeteilt. Index- und Suchfunktionen werden über den Cluster gestartet und auf die jeweiligen Nodes verteilt.
Es besteht auch die Möglichkeit mithilfe einer Replikation die Daten auf einen weiteren Cluster zu übertragen. Dadurch kann bei einem kompletten Ausfall eines Clusters der verbleibende die Funktionen übernehmen.

Document

Elasticsearch speichert die Informationen nicht wie in herkömmlichen Datenbanken als Spalten / Zeilen Kombination, sondern es werden sogenannte Documents abgelegt, welche die Daten als serialisiertes JSON-Format beinhalten. Diese Documents werden auf die Nodes innerhalb eines Clusters verteilt. Der Inhalt dieser wird indexiert und ist dadurch in nahezu Echtzeit abrufbar.

Index

Ein Index kann vereinfacht als eine optimierte Zusammenfassung von mehreren Documents betrachtet werden. Die Documents beinhalten die Key-Value-Paare mit den gespeicherten Daten. Standardmäßig werden

von Elasticsearch alle Felder je nach Datentyp automatisch indexiert. Text-Informationen werden z.B. als invertierter Index gespeichert. Dadurch ermöglicht Elasticsearch es, eine performante Full-Text-Suche anzubieten. Ein weiteres Beispiel sind geographische Informationen, welche in einem BKD-Tree [8] gespeichert werden.

Datentypen

Es werden von Elasticsearch verschiedene Datentypen angeboten, welche teilweise automatisch bei der Indexierung erkannt oder auch manuell im Schema-Mapping definiert werden können. Beispiele für unterstützte Datentypen sind:

Verbreitete Datentypen

- String
- Date
- Numeric (long, integer, short, byte, double und float)
- Boolean
- Binary

Spezielle Datentypen

Elasticsearch bietet ebenfalls Datentypen, welche für spezielle Anwendungsfälle verwendet werden können und hierfür die Verwendung von anwendungsspezifischen Funktionen ermöglicht. Unteranderem besteht die Möglichkeit den Datentyp als „geo-point“ zu definieren. Dadurch kann die geographische Breite und geographische Länge gespeichert werden. Eine statische Definition des Datentyps ermöglicht es, in weiterer Folge die datentypspezifischen Funktionen zu verwenden. In dem Beispiel des geo-point können z.B. Distanzberechnung oder auch die Suche nach allen Koordinaten in einem bestimmten Bereich vorgenommen werden.

Eine weitere Definition, welche in dem Anwendungsfall des Tactical-SIEM einen Anwendungsfall findet, ist die Definition von IP-Adressen. Durch den Datentyp „ip“ kann ein Key-Value-Paar als IP-Adresse definiert werden. Dies ermöglicht es, später bei einer Suche auch die typischen IP-Eigenschaften zu verwenden. Somit kann eine Suche auch per Netz-ID (z.B. 10.0.0.0/24), Wildcard (10.0.0.*) oder auch als Range (10.0.0.0 – 10.0.0.254) durchgeführt werden.

Index-Schema

Da eine automatische Erkennung der Datentypen nicht immer ausreichend bzw. korrekt ist, bietet Elasticsearch die Möglichkeit, ein Schema für die Datentypen anzugeben. In diesem Schema wird eine Zuweisung der zu erwartenden Felder zu einem Datentyp angegeben. Ein Anwendungsbeispiel für eine solche Zuweisung sind die Datentypen Text und Keyword. Bei einer automatischen Erkennung wird ein String als Text erkannt und die einzelnen Wörter indexiert. Durch den Datentyp Keyword kann definiert werden, dass der enthaltene Text als ganzes indexiert und nicht in die einzelnen Wörter unterteilt wird.
Diese unterschiedliche Indexierung hat eine wesentliche Auswirkung auf die retournierten Ergebnisse einer durchgeführten Suchabfrage. So werden bei einer Suche nach einem Wort in einem Keyword indizierten Feld nur Ergebnisse geliefert, bei welchem der gesuchte Text mit dem Text im Feld vollständig übereinstimmen. Bei einem als Text indizierten Feld würden in diesem Fall auch Dokumente retourniert werden, bei welchen das gesuchte Wort nur ein Teil des Inhalts ist.
Ein weiterer Grund für die Keyword-Indizierung ist, dass eine Aggregationsfunktion nicht auf Text-Felder ausgeführt werden kann. Falls eine Aggregation oder Visualisierung der Daten notwendig ist, muss das Feld als Keyword indiziert sein. Das Listing 2.1 zeigt ein Beispiel eines Schema-Mappings, in welchem zwei Textfelder als Keyword und Text definiert werden.

```
{
  "mappings": {
    "_doc": {
      "properties": {
        "email": {
          "type": "keyword"
        },
        "raw_log_message": {
          "type": "text"
        }
      }
    }
  }
}
```

Listing 2.1: Schema Mapping String

Diese unterschiedliche Funktionsweise bzw. Einschränkungen der Datentypen macht es erforderlich, dass

je nach Anwendungsfall bestimmte Datentypen zu den Feldern zugewiesen werden. Damit es durch die Datentypzuweisung zu keinem gegenseitigen Ausschluss von Funktionen kommt, können die Felder auch mehrmals zugewiesen werden. In dem Fall eines Strings für die gleichzeitige Speicherung als Text und Keyword kann das Mapping im Listing 2.2 verwendet werden.

```
{
  "mappings": {
    "_doc": {
      "properties": {
        "name": {
          "type": "text",
          "fields": {
            "keyword": {
            "type": "keyword"
            }
          }
        }
      }
    }
  }
}
```

Listing 2.2: Schema Mapping Text und Keyword

Durch diese Zuweisung wird der Inhalt als Text im Feld „Name“ abgespeichert. Für die Aggregationsfunktionen kann das neu erstellte Feld „Name.Keyword“ als Typ Keyword verwendet werden.

Änderung von Datentypen

Eine nachträgliche Änderung der gespeicherten Datentypen ist nicht möglich. Ist es notwendig, dass die Datentyp Zuweisung geändert wird, muss ein neuer Index mit den gewünschten Datentypen angelegt werden. Dies muss bei der Planung und Implementierung des ELK-Stacks bereits berücksichtigt werden. Wesentliche Faktoren hierfür sind:

- Wie oft wird ein neuer Index erstellt?
- Wie lange müssen die Daten gespeichert und auswertbar sein?

Für die Implementierung des Tactical-SIEM wird nach spätestens 24 Stunden ein neuer Index für die unterschiedlichen Mandanten und Log-Typen erstellt. Eine Änderung eines Datentyps würde somit automatisch auf die neu gespeicherten Daten angewandt werden. Die bereits gespeicherten Daten werden in diesem Fall nicht geändert. Zu diesem Zeitpunkt ist diese Variante für die geplante Umsetzung des Tactical-SIEMs ausreichend. Die Tactical-SIEM-Implementierung analysiert ausschließlich neue Daten und wird mit einer kurzen Aufbewahrungsdauer implementiert. Ist eine Auswertung von älteren Daten notwendig, muss dies auf dem geplanten Compliance-SIEM durchgeführt werden. Sollte sich der Anwendungsfall ändern und eine längere Speicherung der Daten vorausgesetzt werden, wäre es notwendig, die bereits gespeicherten Daten neu zu indexieren. Eine Möglichkeit hierfür wäre, die Daten erneut vom Compliance-SIEM in das Tactical-SIEM zu übertragen. In diesem Fall werden die Daten in einem neuen Index mit der neuen Datentypzuweisung gespeichert.
Eine weitere Variante wäre die Verwendung der Re-Index-API, welche von Elasticsearch angeboten wird. Mithilfe dieser kann ein bestehender Index neu indexiert werden. Die Voraussetzung hierfür ist, dass die Daten in einen neuen Index übertragen werden müssen und somit kurzzeitig den doppelten Speicherplatz benötigen. Ein möglicher Aufruf der Elasticsearch-API wird im Listing 2.3 gezeigt.

```
POST _reindex
{
  "source": {
    "index": "my-index-000001"
  },
  "dest": {
    "index": "my-new-index-000001"
  }
}
```

Listing 2.3: Elasticsearch Reindex API

Shard

Elasticsearch bietet die Möglichkeit, die erstellten Indizes in mehrere Teile aufzuteilen, welche als Shards bezeichnet werden. Diese Shards sind voll funktionsfähige und eigenständige Indizes. Dadurch können die Shards horizontal auf mehrere Nodes verteilt und Datenoperationen parallel durchgeführt werden. Durch diese Funktionalität kann die Performance von Operationen gesteigert werden. Zusätzlich ermöglicht diese

Funktionalität es auch, zu späteren Zeitpunkten Anpassungen bei der Shard-Anzahl durchzuführen. Diese Änderungen werden ebenfalls nur auf neu erstellte Indizes angewandt. Es ist nicht möglich eine global gültige Anzahl an Shards als Optimalwert festzulegen. Bei der Auswahl der Shard-Anzahl muss der Speicherplatz eines Index / Shard und die enthaltenen Dokumente berücksichtigt werden. Wählt man die Shard-Anzahl zu groß oder zu gering kann dies die Leistung bei Datenoperationen verschlechtern. Somit ist diese Konfiguration immer von der jeweiligen Umgebung abhängig und kann im Optimalfall mit Echtdaten getestet werden. Die Abbildung 2.1 zeigt eine Index-Aufteilung bei einer konfigurierten Anzahl von vier Shards und einem Elasticsearch-Cluster bestehend aus zwei Nodes.

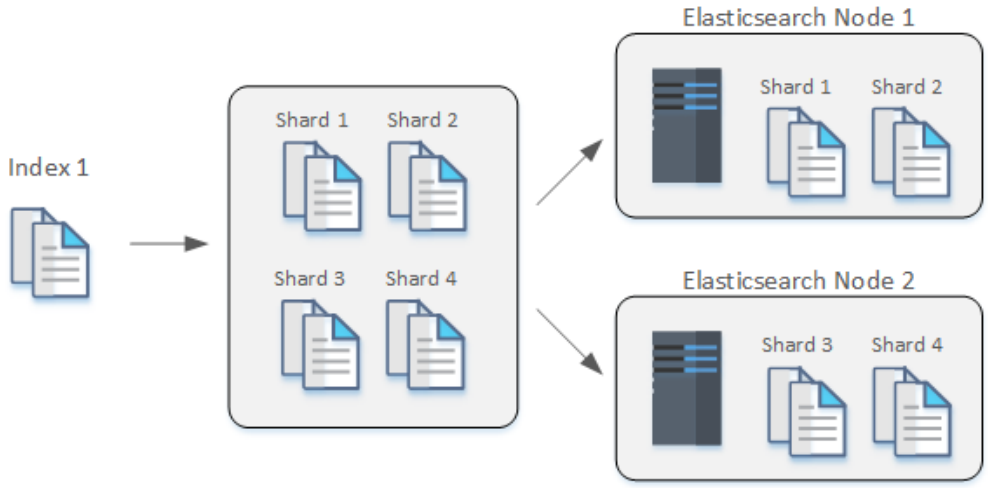

Abbildung 2.1.: Aufteilung von Shards

Replicas

Bei Replicas handelt es sich um eine spezielle Form von Shards, welche auch als „replicas shards“ bezeichnet werden. Diese Funktion wird genutzt um eine Hochverfügbarkeit der Daten zu gewährleisten. Die „replicas shards“ sind Kopien der zuvor erstellten Shards und werden in einem Elasticsearch-Cluster immer auf einem anderem Node als die Produktivdaten abgelegt. Diese Funktion soll sicherstellen, dass auch bei einem Ausfall einer einzelnen Node die Funktionalität des Cluster nicht beeinflusst wird. Folgende Abbildung 2.2 zeigt die Erweiterung der Aufteilung aus Abbildung 2.1 mit einer konfigurierten Anzahl von einem Replika.

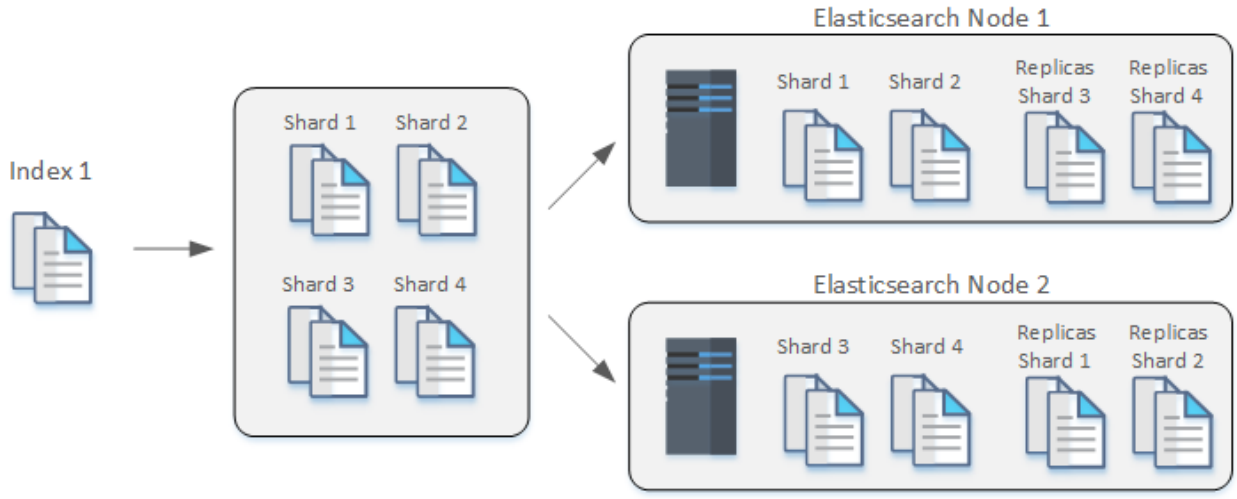

Abbildung 2.2.: Aufteilung von Replicas Shards

Durch diese Aufteilung würde bei einem Ausfall einer Node die jeweils verbleibende Node die Replika-Shard aktivieren und zur Verfügung stellen.

2.3.2. Kibana

Kibana dient zur Visualisierung der Daten und besitzt eine browserbasierte Benutzeroberfläche. Durch Kibana können die aufgezeichneten Log-Daten auf unterschiedliche Varianten dargestellt werden. Es kann aus einer Vielzahl an unterschiedlichen Visualisierungsvarianten wie Histogramme, Graphen oder Diagramme gewählt werden. Zusätzlich können auch direkt die Rohdaten abgefragt und angezeigt werden.

2.3.3. Logstash

Logstash ist eine Softwarekomponente, welche für das Sammeln, Verarbeiten und Weiterleiten von Log-Daten verwendet wird. Die Funktionalität ist hierbei in die Bereiche Input, Filter und Output aufgeteilt. Der Input-Bereich ist für die Sammlung der Daten verantwortlich. Nachdem die Daten angenommen wurden, werden diese in den Filter-Bereich weitergeleitet. In diesem Bereich können die empfangenen Daten weiter angepasst, erweitert oder gelöscht werden. Sobald die Verarbeitung der Daten abgeschlossen ist, kann im Output-Bereich definiert werden, an welches Ziel die Daten weitergeleitet werden sollen [9]. Die Abbildung 2.3 zeigt den Aufbau der einzelnen Logstash-Komponenten.

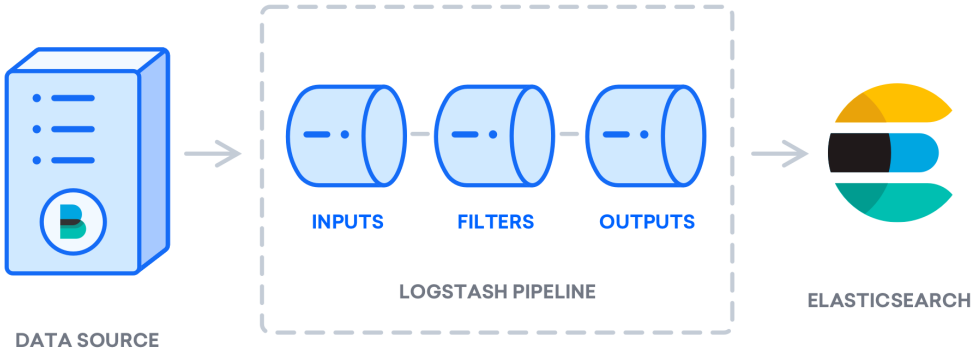

Abbildung 2.3.: Aufbau von Logstash [10]

Durch die bereits vorhandenen Input-Plugins können viele unterschiedliche Protokolle wie syslog, http oder auch die Ausgabe eines weiteren Programms auf der Befehlszeile als Eingabe verwendet werden. Insgesamt stehen zurzeit mehr als 50 unterschiedliche Input-Plugins, welche auch offiziell unterstützt werden, zur Verfügung. Durch diese breite Masse an Input-Möglichkeiten besteht eine hohe Chance, dass ein bestimmtes System direkt mit den vorhanden Mitteln angebunden werden kann [11].

2.3.4. Docker-Container

Als Implementierungsvariante wurden Docker-Container für die einzelnen Systeme gewählt. Die Hauptgründe für diese Entscheidung umfassen unter anderem die einfache Skalierbarkeit, da zusätzliche Container auch auf bereits vorhandenen Systemen implementiert werden können. Ein weiterer wesentlicher Faktor sind die Kompatibilitätsanforderungen der Applikationen. Da das gesamte System aus vielen unterschiedlichen Komponenten besteht, ist es wesentlich, dass diese auch unabhängig von einer einzigen funktionieren. Dies gilt ebenso für Aktualisierungen der einzelnen Komponenten. Durch die Docker-Variante können einzelne Container sehr einfach mit einer neueren Version des Containers ausgetauscht werden. Sollte es mit dieser Version Probleme geben, kann auch wieder die alte Version gestartet werden. Diese Vorgehensweise erleichtert die Wartungsarbeiten und die möglicherweise notwendige Erweiterung des Systems [12].
Da das System ebenfalls für zukünftige Erweiterungen ausgelegt sein sollte, bieten sich hier ebenfalls Docker-Container an. Diese ermöglichen es, dass neue Container mit bestimmten Funktionen auf die bereits vorhandenen Server hinzugefügt werden. Bei diesem Vorgang muss wiederum nicht auf den bereits bestehenden Systemen bzw. deren installierten Komponenten Rücksicht genommen werden. Nachteile der Docker-Installationsvariante sind der zusätzliche Aufwand für die Implementierung und die Abhängigkeiten von den unterschiedlichen Konfigurationsdateien. Da dieses System aber vollständig mithilfe von Ansible-Playbooks implementiert wird, sollte dies für zukünftige Arbeiten keine Auswirkungen haben, da die initiale

Logik bereits in einem Playbook beschrieben ist. Die Abbildung 2.4 zeigt eine Übersicht der geplanten Architektur.

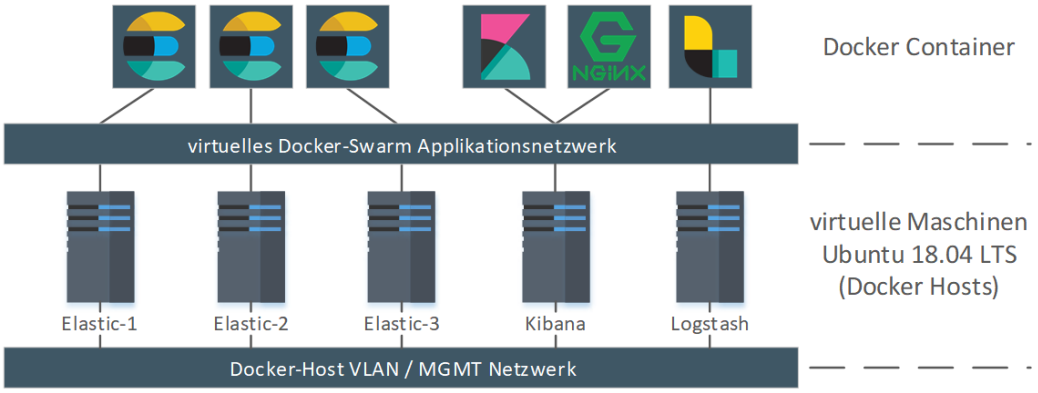

Abbildung 2.4.: Überblick Docker-Container

Docker-Swarm-Overlay-Network

Das Docker-Swarm-Overlay-Network erstellt ein virtuelles Netzwerk über die Docker-Host Mitglieder, welches für die weitere Kommunikation zwischen den Docker-Container verwendet wird. Das Overlay-Network bietet den Vorteil, dass das Routing bzw. die Weiterleitung der Pakete automatisch durch Docker gehandhabt wird. Ein weiterer Vorteil dieser Variante ist, dass die übertragenen Pakete innerhalb des Overlay-Network verschlüsselt werden können [13].

2.4. Open-Source-Threat-Intelligence

Unter dem Begriff Threat-Intelligence werden Kenntnisse über Angriffe oder schadhaften Systeme zusammengefasst. So werden mithilfe von Threat-Intelligence Informationen wie Vorgehensweise, Methode und die Indicator-of-Compromise (IOC) dokumentiert und für weitere Analysen aufbereitet. Diese Informationen können unter anderem aus Sandbox- oder Honeypot-Systemen extrahiert werden. Zusätzlich werden diese Informationen auch aus tatsächlichen Angriffen extrahiert und sollen bei der Erkennung und Verhinderung eines weiteren Angriffs beitragen. Der Mehrwert solcher Threat-Intelligence-Informationen ist, dass bereits bekannte Angriffe verhindert oder frühzeitig erkannt werden können.
Die ständig wachsenden Angriffsmöglichkeiten und deren Komplexität erfordern es, dass solche Informationen automatisch extrahiert und ausgetauscht werden können. Durch diese Anforderungen wurden Standards wie Structured-Threat-Information-eXpression (STIX) für die Erfassung der Cyber-Threat-Intelligence und Trusted-Automated-eXchange-of-Intelligence-Information (TAXII) für den Austausch dieser Informationen

definiert [3][14].
Aufgrund der großen Menge an vorhandenen Informationen stellen die Verwaltung und eine einfache Verwendung dieser ein weiteres Problem dar. Durch diese Herausforderung wurde unter anderem das Projekt Malware-Information-Sharing-Platform-and-Threat-Sharing (MISP) [15] gestartet.

2.4.1. Anforderungen an Cyber-Threat-Intelligence

Durch die Integration von Open-Source-Cyber-Threat-Intelligence-Lösungen sollen die empfangen Daten mit bereits bekannten und als schadhaft eingestuften Informationen verglichen werden. Folgende Informationen wurden gewählt, da diese für einen Vergleich mit typischen Details in Log-Einträgen verwendet werden können.

IP-Adressen

Traffic-Logs von Firewalls beinhalten die Source- und Destination-IP Adressen einer Verbindung. Durch öffentliche Listen, die IP-Adressen welche durch ein schadhaftes Verhalten aufgefallen sind umfassen, können diese Informationen für eine Anreicherung verwendet werden. Zusätzlich kann diese Funktion auch für den Abgleich mit bestimmen Services wie z.B. TOR verwendet werden.
Das Github-Projekt „IPSum“ [16] beinhaltet mehrere Listen, welche aus unterschiedlichen öffentlichen Blacklisten aggregiert werden. Hierfür werden mehr als 30 unterschiedlichen Quellen täglich abgerufen und zusammengeführt. Das Projekt bietet mehrere Listen an, welche in Level eingestuft sind. Die Level beschreiben, auf wie vielen Blacklisten die enthaltenen IP-Adressen gefunden wurden. Die „IPSum Level 3“ Liste beinhaltet z.B. nur IP-Adressen, welche in mindestens drei unterschiedlichen Quellen gefunden wurden. Diese Einstufung kann verwendet werden, um die False-Positive Rate der IP-Adressen zu reduzieren.

Domänen

Wie bei den IP-Adressen gibt es bereits vorhandene Listen von Domänen, welche direkt abgerufen werden können. Diese Domänen können wiederum für den Abgleich von Webfilter, Proxy-Logs oder auch DNS-Logs verwendet werden. Auch für diesen Anwendungsfall gibt es ein Github-Projekt, welches eine aggregierte Liste aus mehreren frei zugänglichen Blacklists zur Verfügung stellt. Das Projekt „Blackbook“ [17] führt hierbei folgende verfügbaren Listen zusammen:

- CyberCrime
- URLhaus
- ScumBots

- Benkow
- VirusTracker

Das Ziel des Projektes ist es, dass eine Liste von Domänen zur Verfügung gestellt wird, welche aktiv für Malware verwendet wurden oder noch aktiv verwendet werden. Der Verwendungszweck der Domänen ist in den meisten Fällen für eine Command-and-Control Verbindung vorgesehen. Die Liste beinhaltet keine kompromittierten Domänen, welche z.B. für die Malware-Verteilung verwendet werden. Diese Liste soll dazu dienen, dass infizierte Clients einer Umgebung identifiziert werden können.

Hashwerte

Es besteht die Möglichkeit, die Hashwerte von ausgeführten Dateien zu protokollieren. In einer Windows-Umgebung kann dies z.B. mit dem Tool Sysmon bewerkstelligt werden. Zusätzlich bieten auch Antivirus-Softwarehersteller solche Möglichkeiten an.
Diese protokollierten Hashwerte können wiederum mit Hashwerten von bereits bekannten schadhaften Dateien verglichen werden. Listen mit schadhaften Datei-Hashwerten können hauptsächlich durch kommerzielle Lösungen bezogen werden.
Beispiele für solche Bezugsquellen von Hashwerten sind:

- Kaspersky-Threat-Data-Feeds (Demo-Version liefert die ersten 100 Hashes) [18]
- Cyber-Cure (Freie Version ist verfügbar aber auf 100 Hashes limitiert) [19]
- VirusTotal-Premium-Services (Keine freie Version für Hashwerte vorhanden) [20]

2.5. Open-Source-SIEM-Regelwerk

Damit ein SIEM-Regelwerk austauschbar gestaltet werden kann, ist es notwendig, dass die Definition der Regeln herstellerunabhängig und standardisiert ist. Für die Verwendung des Regelwerks muss eine Möglichkeit geboten werden, die Regeldefinitionen auf ein Zielsystem zu konvertieren. Ein verbreitetes Open-Source-Projekt, welches diese Eigenschaften erfüllt, ist Sigma [21].
Bei Sigma handelt es sich um ein generisches Format für die Beschreibung von SIEM-Regeln. Dadurch soll erreicht werden, dass die unterschiedlichen Regeln einfach beschrieben und ausgetauscht werden können. Beim Entwurf des Schemas wurde deswegen auch darauf geachtet, dass das Format flexibel und strukturiert ist. Für die Verwendung der Regeln ist es notwendig, diese für das gewünschte Zielsystem zu konvertieren. In diesem Schritt werden auch die notwendigen Logquellen bzw. Felder zugewiesen.

2.5.1. Aufbau des Formats

Die Sigma-Regeln werden im YAML-Format beschrieben und können die im Listing 2.4 gezeigten Felder beinhalten.

```
title
id [optional]
related [optional]
type {type-identifier}
id {rule-id}
status [optional]
description [optional]
author [optional]
references [optional]
logsource
category [optional]
product [optional]
service [optional]
definition [optional]
...
detection
{search-identifier} [optional]
{string-list} [optional]
{field: value} [optional]
...
timeframe [optional]
condition
fields [optional]
falsepositives [optional]
level [optional]
tags [optional]
...
[arbitrary custom fields]
```

Listing 2.4: Sigma Format

Für eine Sigma-Regel werden mindestens die Felder Title, Logsource und Detection vorausgesetzt.

Title

Ein kurzer Titel der beschreiben soll, was diese Regel entdecken soll. Dieses Feld darf maximal 256 Zeichen beinhalten.

Logsource

Dieses Attribut beschreibt die Logquelle, auf welche die definierte Regel angewandt werden soll. Für die Beschreibung der Logquelle beinhaltet das Attribut drei weitere Attribute:

Category:
Dieses Feld wird verwendet, um zu definieren, von welcher Gruppe von Geräten oder Software (z.B. Firewall, Antivirus), die notwendigen Event-Daten erzeugt werden. Für Windows-Eventlog-Regeln wird diese Feld auch für bestimmte Funktionskategorien verwendet (z.B. Prozess-Start, Netzwerkzugriffe)

Product:
Das Feld „Product“ definiert bestimmte Produkte als Logquellen (z.B. Windows, Apache). Bei der Definition Windows würde dies z.B. bedeuten, dass der gesamte Windows-Eventlog verwendet wird (z.B. Security und System)

Service:
Das Feld „Service“ schränkt die Logquellen auf bestimme Services ein (z.B. sshd, sysmon). Bei Produkten wie Windows kann mithilfe des Service-Feldes auch der Windows-Eventlog auf die einzelnen Kategorien (Security, System) unterschieden werden.

Detection

Bei dem Inhalt des „Detection“ Feldes handelt es sich um den wesentlichen Informationsgehalt der Sigma-Regel. In diesem Feld wird beschrieben, mithilfe welcher Suchabfrage die definierte Anomalie entdeckt werden kann. Die Abfrage kann auf mehrere Arten aufgebaut sein. Einerseits kann das Feld eine einzelne Bedingung enthalten. Es besteht auch die Möglichkeit mehrere Abfragen mit einem Logikoperator in diesem Feld zu speichern. Somit können mehrere Suchabfragen mit einer ODER- bzw. einer UND-Verknüpfung kombiniert werden. Zusätzlich können die Bedingungen auch verschachtelt gestaltet werden. Für die einzelnen Definitionen der Suchabfragen stehen unter anderem folgende Funktionen zur Verfügung:

- Wildcard-Suche
- contains, endswith, startswith
- Reguläre-Ausdrücke
- Aggregationsfunktionen (count, min, max, avg und sum)

Ein Beispiel für eine Sigma-Regeln mit mehreren möglichen Suchabfragen ist z.B. die Regel „Maze Ransomware“, welche die gleichnamige Ransomware anhand der Prozesscharakteristik erkennen soll. Die „Detection“ Definition für diese Regel ist im Listing 2.5 beschrieben.

```
detection:
  # Dropper
  selection1:
  ParentImage|endswith:
  - '\WINWORD.exe'
  Image|endswith:
  - '*.tmp'
  # Binary Execution
  selection2:
  Image|endswith: '\wmic.exe'
  ParentImage|contains: '\Temp\'
  CommandLine|endswith: 'shadowcopy delete'
  # Specific Pattern
  selection3:
  CommandLine|endswith: 'shadowcopy delete'
  CommandLine|contains: '\..\..\system32'
  condition: 1 of them
```

Listing 2.5: Sigma Regel Detection – Maze Ransomware

In dieser Regel werden mehrere Suchabfragen (selection1 – selection 3) angegeben. Jede einzelne davon erkennt ein vermutlich schadhaftes Verhalten, weswegen diese durch eine ODER-Verknüpfung kombiniert wurden (condition: 1 of them). Somit würde diese Regel bei nur einem positiven Ergebnis einer dieser Abfragen ausgelöst werden.

2.6. MITRE-ATT&CK-Matrix

Die MITRE-ATT&CK-Matrix (MITRE's Adversarial Tactics, Techniques, and Common Knowledge) ist ein Modell zur Beschreibung des Verhaltens eines Angriffs. Das Modell beinhaltet die unterschiedlichen Phasen eines Angriffes und beschreibt die einzelnen Angriffsschritte mit deren gewünschtem Ziel. Die erste Version des ATT&CK-Modells wurde 2013 begonnen und fokussierte sich auf Angriffe innerhalb einer Windows-Umgebung. Das Modell wurde mithilfe von internen Forschungsprojekten weiter überarbeitet und 2015 veröffentlicht. Zu diesem Zeitpunkt beinhaltete das Modell 96 Angriffstechniken unterteilt in neun Taktiken. Die aktuelle Version des ATT&CK-Enterprise-Modell beinhaltet mittlerweile 330 Angriffstechniken, welche in zwölf Taktiken unterteilt sind. Zusätzlich bildet das Modell in der derzeitigen Ausbaustufe Angriffstechniken für folgende Systeme:

- Windows
- macOS
- Linux
- Cloud

Neben dem ATT&CK-Enterprise-Modell wird ebenfalls eine ATT&CK-Mobile-Version zur Verfügung gestellt. Dieses Modell beinhaltet die relevanten Angriffe und Methode für Mobilgeräte (Android, iOS).

2.6.1. Idee und Aufbau der Matrix

Ein Eintrag in der ATT&CK-Matrix wird in mehrere Teile unterteilt.

Technology Domain

Die MITRE-ATT&CK-Matrix wird im ersten Schritt in „technology domains“ unterteilt. Innerhalb jeder Technology-Domain sind mehrere Plattformen definiert. Bei einer Plattform handelt es sich um das betroffene System, welches mithilfe einer Angriffstechnik attackiert wird. Mit dem Plattformwert wird definiert, für welches Betriebssystem die dokumentierten Angriffstechnologien angewandt werden können. Die Tabelle 2.1 zeigt die Zuweisung der Plattformen.

Technologie Domän	Definierte Plattformen
Enterprise	Linux, macOS, Windows, Cloud
Mobile	Android, iOS

Tabelle 2.1.: MITRE ATT&CK Matrix Domänen

Taktik

Die Taktik beschreibt, warum eine bestimmte Angriffstechnik angewandt wird bzw. welches Ziel mit dieser Technik erreicht werden soll. In der Matrix wird jede Taktik als Spalte dargestellt. Zum derzeitigen Zeitpunkt sind folgende zwölf Taktiken in der Matrix definiert:

- Initial Access
- Execution
- Persistence
- Priviledge Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact

Technik

Die Technik beinhaltet, welche Aktionen ausgeführt werden können, damit ein definiertes Ziel erreicht wird. Zusätzlich kann die Technik auch eine Beschreibung beinhalten, welche Informationen durch diese ermittelt werden können. Dies ist vor allem bei Taktikkategorien wie „Collection“ oder „Exfiltration“ relevant. Die jeweilige Technik wird in der Matrix in einer Taktikspalte platziert, wobei eine Technik in mehreren Spalten angesiedelt sein kann. Ein Beispiel hierfür ist die Technik „Hooking“. Diese kann für „Priviledge Escalation“ und „Credential Access“ verwendet werden. Für die weitere Verwendung und Verknüpfung von Techniken werden diesen eindeutigen Identifikator zugewiesen. Diese Technik-ID wird wie folgt dargestellt:

T[technique]
Bsp.: T1134 (Access-Token-Manipulation)

Aufbau des Technik-Objekts

Ein Technik-Objekt in der MITRE-ATT&CK-Matrix-Enterprise beinhaltet mehrere Felder des Type Tag oder Field. Das Tag-Feld wird für Referenzen verwendet, damit der Informationsgehalt der Technik einfach erweitert und für Filter- und Auswahlmöglichkeiten verwendet werden kann. Die Tags sind hierbei bereits vorhandene Informationen, welche bei der Technik hinterlegt werden. Beispiel für solche Tags sind z.B. Taktik oder Plattform. Der Type „Field“ wird als Freitextfeld verwendet und beinhaltet spezifische Informationen und Details über die beschriebene Technik. In der Tabelle 2.2 wird ersichtlich, welche definierten Felder eine Technik besitzen kann. Bei den durch „*“ gekennzeichneten Felder handelt es sich um Pflichtangaben.

Feld	**Type**
Name*	Field
ID*	Tag
Tactic*	Tag
Description*	Field
Platform*	Tag
System Requirements	Field
Permission Required*	Tag
Effective Permission*	Tag
Data Source*	Tag
Support Remote	Tag
Defense Bypassed	Tag
CAPEC ID	Field
Contributor	Tag
Examples	Relationship / Field
Detection*	Tag
Mitigation*	Tag

Tabelle 2.2.: MITRE-ATT&CK-Matrix-Technik-Felder

Sub-Technik

Aufgrund des ständig wachsenden Umfangs der MITRE-ATT&CK-Matrix wurde eine weitere Unterteilung der Techniken vorgestellt. In Zukunft sollen die Techniken zusätzlich in Sub-Techniken unterteilt werden. Dies bewirkt, dass die einzelnen Techniken noch genauer unterteilt und besser strukturiert werden können.

Ein Beispiel hierfür bietet z.B. die bereits bekannte Technik „Access Token Manipulation“. Bei dieser Technik wurde bisher nicht unterschieden, ob ein bestehender Token übernommen oder ob ein neuer Token erstellt wurde. Mithilfe der neuen Sub-Techniken kann diese Technik genauer definiert werden. In der derzeitigen Implementierung sind folgende Sub-Techniken der Technik Access-Token-Manipulation zugeteilt:

- Token Impersonation / Theft
- Create Process with Token
- Make an Impersonate Token

Durch die neuen Sub-Techniken wird ebenfalls die Identifikation der Techniken erweitert. Es werden hierfür die vorhandenen Technik-IDs wie folgt ergänzt.

T[technique].**[sub-technique]**

Für das oben genannte Beispiel würde dies bedeuten, dass die Technik „Access Token Manipulation“ weiterhin die ID T1134 verwendet. Die Sub-Technik „Token Impersonation / Theft“ würde die ID T1134.001 erhalten.

Groups

Eine Gruppe ist eine benannte Zusammenfassung aus mehreren relevanten Informationen, welche für ein bestimmte Angriffsbeschreibung hilfreich sind. Eine Gruppe wird in den meisten Fällen aufgrund folgender Informationen gebildet:

- Aktoren, welche die Angriffe durchgeführt haben
- Angriffstyp
- Kampagnen in welchen die Techniken oder Software verwendet wurde.

Die jeweiligen Gruppen können in der ATT&CK-Matrix untereinander verknüpft werden. Dies bietet den Vorteil, dass z.B. für einen gezielten Angriff eine Gruppe für die Angriffskampagne erstellt und diese z.B. mit einer Gruppe der durchführenden Aktoren verknüpft werden kann. Somit können bereits vorhandene Informationen verknüpft und müssen nicht redundant gepflegt werden.

Software

Die Software einer MITRE-ATT&CK-Matrix gibt das eingesetzte Programm für die erfolgreiche Durchführung der Technik an. Es gibt für die Software keine Unterscheidung, ob es sich um eine Open-Source-Lösung oder ein kommerzielles Programm handelt. Zusätzlich kann eine angegebene Software auch ein Betriebssystemdienst sein. Folgende Aufzählung zeigt mögliche Beispiele für den Software-Eintrag:

- Tasklist (Standardprogramm für die Anzeige von aktiven Prozessen in Windows-Betriebssystemen)
- Mimikatz (Sammlung und Verwendung von Windows-Anmeldeinformationen)
- Cobalt Strike (Framework für die Ausführung von Exploits, Metasploit und Mimikatz-Integrationsmöglichkeiten)
- OwaAuth (Web-Shell / Malware ausschließlich für Microsoft Exchange Server)

Die Software-Einträge in der Matrix können ebenfalls mit Gruppen verknüpft werden. Dadurch ist es möglich, dass für eine bestimmte Gruppe die gesamte eingesetzte Software ermittelt werden kann.

2.6.2. Vergleichbare Projekte

Ein auf den ersten Blick sehr ähnliches Projekt ist Common-Attack-Pattern-Enumeration-and-Classification (CAPEC). Die beiden Projekte ATT&CK und CAPEC fokussieren sich jedoch auf jeweils unterschiedliche Schwerpunkte. CAPEC spezialisiert sich primär auf Applikationssicherheit und bekannte Schwachstellen für angreifbare Ziele. In diesem Projekt werden bestimmte Angriffsmuster für die Ausnutzung von Schwachstellen beschrieben. Die CAPEC-Informationen werden auch mit der Common-Weakness-Enumeration (CWE) verknüpft. ATT&CK fokussiert sich hingegen auf die Netzwerksicherheit und den gesamten Lebenszyklus von Advanced-Persistent-Threats. So werden in den Taktiken und Techniken auch die Phasen für die Vorbereitung, die Durchführung wie auch die Tätigkeiten nach einem erfolgreichen Angriff beschrieben. Zusätzlich beinhaltet die ATT&CK-Matrix auch eine Beschreibung für gewünschte Ergebnisse eines dokumentieren Angriffs. Der Informationsinhalt der Matrix ist hierbei von Threat-Intelligence und IT-Security Forscher/innen abhängig [22].
Die Informationen aus den Projekten werden im jeweils anderen Projekt verwendet, um den Informationsgehalt zu verbessern. Somit ist es möglich, dass ein CAPEC-Angriffsmuster mit einer ATT&CK-Matrix-Technik verknüpft wird, damit ein besseres Verständnis für das Angriffsmuster geschaffen wird.

2.6.3. Anwendungsbereiche

Die MITRE-ATT&CK-Matrix soll eine Übersicht der möglichen Angriffstechniken mit detaillierter Beschreibung bereitstellen. Diese Informationen sollen für unterschiedliche Anwendungsbereiche hilfreich sein [23].

Blue-Team

Mithilfe der ATT&CK-Matrix kann ermittelt werden, welche Techniken die bereits umgesetzten Maßnahmen erkennen bzw. verhindern können. Durch diese Information können gezielt weitere Maßnahmen geplant und verbessert werden. Ein weiterer Anwendungsfall ist eine gezielte Mitigation oder Erkennung eines speziellen Angriffs. In der ATT&CK-Technik werden ebenfalls Mitigationen angegeben, welche die erfolgreiche Durchführung verhindern sollen. Zusätzlich werden Indikatoren beschrieben, welche auf die Ausführung der Angriffstechnik hinweisen [24]. Im Fall eines erkannten Angriffs können weitere Informationen wie z.B. die durchführende Gruppe eines Angriffs und deren weiteren Techniken über die Matrix ermittelt werden.
Ein Beispiel hierfür ist die oben genannte Software „OwaAuth“. Hier kann über die ATT&CK-Matrix ermittelt werden, dass diese ausschließlich von der Gruppe „Threat Group-3390“ verwendet wurde. Mithilfe der Matrix kann nun kontrolliert werden, welche weiteren Techniken oder Programme von dieser Gruppe genutzt und gezielt nach diesen Indikatoren gesucht werden.

Red-Team

Bei der Durchführung eines Penetrationstests kann die Matrix verwendet werden, damit unterschiedliche Techniken für ein gewünschtes Ergebnis ermittelt werden. Somit kann z.B. das gewünschte Ergebnis ein „Credential Access“ sein. Durch die Matrix können nun verschiedene Varianten ausgewertet werden. In weiterer Folge würden sich z.B. die Techniken „OS Credential Dumping: LSASS Memory“ oder „Steal or Forge Kerberos Tickets: Kerberoasting“ ergeben. In der jeweiligen Technik befinden sich zusätzlich zu einer Beschreibung der Vorgehensweise auch Beispiele wie diese durchgeführt werden können.

IT-Security-Implementierungen

Wie auch beim Blue-Team kann bei IT-Security-Implementierungen bzw. Softwareprodukten die Matrix verwendet werden, um eine beinahe vollständige Abdeckung der bekannten Angriffstechniken zu erreichen. Für eine erleichterte und besser nachvollziehbare Analyse bei einem erkannten oder verhinderten Angriff werden bereits bei einigen Softwareprodukten die Matrix Technik-IDs in den generierten Log-Einträgen angegeben. Mithilfe dieser kann ein/e Administrator/in der ATT&CK-Matrix die weiteren Informationen wie

das gewünschte Ergebnis des Angriffs kontrollieren. Beispiele für eine Erweiterung des Informationsgehalt anhand einer Sigma-Regel befinden sich im Listing 2.6.

```
tags:
- attack.initial_access
- attack.t1190
```

Listing 2.6: Sigma Tag Feld für ATT&CK Matrix Technik

In einem Sysmon-Regelwerk wird die Technik-ID bei der Generierung des Eventlogs mitangegeben. Im Listing 2.7 befindet sich ein Beispiel anhand einer „UAC Tampering“ Regel.

```
<TargetObject name="T1088" condition="end with">HKLM\Software\
   Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA</
   TargetObject>
```

Listing 2.7: Sysmon Konfiguration mit ATT&CK Matrix Technik

2.7. Relevante Log-Quellen

Bei Windows-Systemen wird die Protokollierung von Diensten und Änderungen am System in den Windows-Eventlog gespeichert. Die generierten Events werden im XML-Format erstellt. Das Windows-Eventlog ist für die allgemeine Protokollierung in folgende Kategorien unterteilt:

- Application
- Security
- System

Zusätzlich bietet das Eventlog weitere Kategorien für spezielle Dienste oder aktivierte Audit-Funktionen. Für die Auswertung, Kontrolle und Alarmierung im geplanten Tactical-SIEM werden eine Vielzahl von Events benötigt. Aus den später verwendeten Sigma-Regeln können folgende Standard Event-Kategorien ausgewertet werden:

- Windows Security
- Windows Applocker
- Windows System

- Windows Application
- Windows Defender
- Windows PowerShell
- Windows PowerShell-Classic

Bei den Sigma-Regeln ist es ebenfalls möglich, dass keine spezielle Log-Quelle angegeben ist. Dies ist z.B. bei den Regeln der Kategorie „process creation“, „process access“ und „image_load“ der Fall. Diese Regeln suchen meist nach bestimmten Namen in dem Windows-Eventlog typischen Feld „ImageName“. Der Grund der fehlenden Definition einer spezielle Kategorie ist, dass es mehrere Möglichkeiten für die Erkennung geben kann. So ist es z.B. möglich, dass Standard Windows-Event mit der ID 4688 (Windows / Security) zu verwenden oder den Prozessstart über ein Sysmon-Event (Sysmon-Event ID 1) zu protokollieren. Bei der Auswertung aller Windows-Sigma-Regeln konnte die Aufteilung auf die unterschiedlichen Kategorien ermittelt werden. Die Tabelle 2.3 zeigt die Aufteilung der Kategorien für alle Sigma-Regeln, welche ein definiertes Service-Feld haben.

Definiertes Service-Feld	**Anzahl**
Windows Security	74
Windows Applocker	1
Windows System	16
Windows Application	5
Windows Defender	2
Windows PowerShell	20
Windows PowerShell-classic	3
Gesamt	121

Tabelle 2.3.: Event-Typ in Windows Sigma Regeln

Eine Auswertung der verwendeten Windows Event IDs befindet sich im Anhang A.2. Die Tabelle 2.4 zeigt die Kategorie-Aufteilung von Sigma-Regeln ohne spezifizierten Service-Feld.

Kategorie	Anzahl	Sysmon EventID
process_creation	277	1
driver_load	1	6
file_event	17	9, 11, 15
image_load	17	7
network_connection	12	3
process_access	7	10
registry_event	36	12, 13, 14
CreateRemoteThread	6	8
PipeEvent	4	17, 18
WmiEvent	3	19, 20, 21
Gesamt	380	-

Tabelle 2.4.: Event-Kategorien in Windows Sigma Regeln

Aus der Auswertung der verwendeten Events in den Sigma-Regeln ist ersichtlich, dass ein Großteil der Definitionen zusätzlichen Eventlog-Quellen benötigt. Damit diese Informationen auf den Systemen gesammelt werden können, müssen die Einstellungen der Protokollierung angepasst und ein zusätzlicher Service (System-Monitor) installiert werden.

2.7.1. PowerShell-Logging

Angriffe auf aktuelle Windows Umgebungen werden immer häufiger über die integrierten PowerShell-Funktionen von Windows-Systemen durchgeführt [25]. Da Ausführungen von Schadcodes mithilfe der PowerShell nur im Arbeitsspeicher stattfinden und keine Dateien erstellt werden müssen, können herkömmliche Antivirus-Programme keinen Schadcode feststellen. Dies ist nur über Antivirus-Software möglich, welche das Windows-Antimalware-Scan-Interface (AMSI) verwendet. Aus diesen Gründen ist eine Protokollierung der ausgeführten Befehle für die Erkennung von schadhaften Verhalten notwendig [26].
Ältere PowerShell-Versionen (2.0, 3.0 und 4.0) protokollieren bei einer Ausführung von PowerShell nur minimale Informationen in dem dazugehörigen Windows-Event. Diese Informationen beinhalten bei der PowerShell Version 2.0 den Start- und Endzeitpunkt der PowerShell-Sitzung und ob es sich um eine lokale Ausführung oder eine PowerShell-Sitzung über das Netzwerk handelt. Es werden in dieser Variante keine Informationen zu den ausgeführten Befehlen oder Skripten protokolliert. Diese Funktionalität wurde in der PowerShell Version 3.0 mit dem Module-Logging erweitert. Auf den aktuellen Windows-Systemen

(Windows 10, Server 2016 und Server 2019) werden, ohne weitere Anpassungen, drei unterschiedlichen Protokollierungsmöglichkeiten der PowerShell unterstützt [27].

- Module Logging (ab PowerShell 3.0)
- Script Block Logging (ab PowerShell 5.0)
- Transcription (ab PowerShell 2.0)

Module-Logging

Das Module-Logging der PowerShell zeichnet die Pipeline-Ausführungsdetails auf. Diese Informationen beinhalten die Befehlsaufrufe und Variableninitialisierungen. Zusätzlich wird die Ausgabe der ausgeführten Befehle protokolliert. Jedoch werden in dieser Variante keine dynamisch generierten Befehle aufgezeichnet. Da die meisten Angriffe über PowerShell mit Encoded-Skript-Blöcken umgesetzt werden, wird im Eventlog nur der Encoded-Wert gespeichert. Eine automatische Erkennung des ausgeführten Befehls ist in diesem Fall nicht möglich. Damit die Module-Logging Funktion aktiviert wird, müssen die Windows-Registry-Schlüssel aus dem Listing 2.8 gesetzt werden oder die entsprechende Gruppenrichtlinien in der Windows-Domäne definiert werden:

```
HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\
PowerShell\ModuleLogging
  EnableModuleLogging = 1
HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\
PowerShell\ModuleLogging\ModuleNames
  * = *
```

Listing 2.8: Module Logging Registry Schlüssel

Durch den Registry-Schlüssel „ModuleNames“ kann die Module-Logging-Funktion auch auf bestimmte Module angewandt werden. Mit dem gesetzten Wert `* = *` werden von allen ausgeführten Modulen alle generierten Events aufgezeichnet.

Script-Block-Logging

In der PowerShell-Version 5.0 wurde die Protokollierung mit dem Script-Block-Logging erweitert. Diese Funktion zeichnet die ausgeführte Befehlsblöcke während der Ausführung auf. Im Fall der Ausführung eines Encoded-Skript-Block würde in dieser Variante der Aufruf mit dem Encoded-Wert protokolliert werden. Zusätzlich wird ein weiteres Event bei der Ausführung des Decoded-Skript-Block generiert. Damit die

Script-Block-Logging-Funktion aktiviert wird, muss der Windows-Registry-Schlüssel aus dem Listing 2.9 gesetzt werden oder die entsprechende Gruppenrichtlinie in der Windows-Domäne definiert werden.

```
HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\
   ScriptBlockLogging
  EnableScriptBlockLogging = 1
```

Listing 2.9: Script Block Logging Registry Schlüssel

Das Listing 2.10 zeigt ein Beispiel für die Ausführung der PowerShell mit einem Encoded-Befehl.

```
powershell -E ZABpAHIAIABjADoAXAA=
```

Listing 2.10: PowerShell Befehl Base64 Encoded

Durch die Ausführung wird das Windows-Event 4104 mit Inhalt aus dem Listing 2.11 erstellt.

```
Creating Scriptblock text (1 of 1):
powershell -E ZABpAHIAIABjADoAXAA=

ScriptBlock ID: 1bb1484d-6377-40f2-9a31-3b5c66b8c89a
Path:
```

Listing 2.11: PowerShell Logging Base64 Encoded

Nachdem der Encoded-Wert dekodiert wurde, wird ein weiteres Event 4104 protokolliert, welches den lesbaren Befehl, wie im Listing 2.12 ersichtlich, beinhaltet.

```
Creating Scriptblock text (1 of 1):
dir c:\

ScriptBlock ID: 20a02204-396b-4a04-8b2e-aa6ccab6f968
Path:
```

Listing 2.12: PowerShell Logging Base64 Decoded

Für die automatische Analyse der PowerShell-Logs bietet das „Script-Block-Logging“ die notwendigen Informationen.

Transcription

Die PowerShell-Transcription-Funktion ermöglicht es, dass alle Eingaben und Ausgaben einer PowerShell-Sitzung in eine definierte Textdatei abgespeichert werden. Die Protokollierung kann über Windows-Registry-

Schlüssel, Gruppenrichtlinien oder pro PowerShell-Sitzung aktiviert werden. Die notwendigen Registry-Schlüssel für die Aktivierung und Konfiguration des Ausgabeordners befindet sich im Listing 2.13.

```
HKLM\Software\Policies\Microsoft\Windows\PowerShell\
   Transcription
  EnableTranscripting = 1
  OutputDirectory = "Pfad"
```

Listing 2.13: PowerShell Transcription Konfiguration

Innerhalb einer PowerShell-Sitzung kann das Transcript-Logging mit dem Befehl aus dem Listing 2.14 gestartet werden.

```
Start-Transcript -Path "Pfad zu Ausgabedatei"
```

Listing 2.14: Transcription innerhalb PowerShell-Sitzung aktivieren

Das Listing 2.15 zeigt die automatisch erstellte Log-Datei. Diese beinhaltet zusätzlich zu den ausgeführten PowerShell-Befehlen und deren Ausgaben auch Informationen über das System, Benutzerinformationen und Versionsinformationen.

```
**********************
Windows PowerShell transcript start
Start time: 20210112211929
Username: DESKTOP-08T8DGN\user
RunAs User: DESKTOP-08T8DGN\user
Configuration Name:
Machine: DESKTOP-08T8DGN (Microsoft Windows NT 10.0.18363.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\
   powershell.exe
Process ID: 9248
PSVersion: 5.1.18362.1171
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.18362.1171
BuildVersion: 10.0.18362.1171
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
```

```
SerializationVersion: 1.1.0.1
**********************
Transcript started, output file is C:\temp\transcript0.txt
PS C:\Users\user> dir c:/

Directory: C:\

Mode     LastWriteTime      Length Name
------   -------------      ------ ----
d-r---   10.01.2021 19:23          Program Files
d-r---   28.12.2020 10:24          Program Files (x86)
d-r---   25.05.2019 21:11          Users
d-----   06.01.2021 17:46          Windows

**********************
Windows PowerShell transcript end
End time: 20210112211952
**********************
```

Listing 2.15: PowerShell Transcription Log-File

Die Funktion des Transcription-Logging der PowerShell kann für forensische Analysen oder für Aufzeichnungen von durchgeführten Tests verwendet werden. Aufgrund der Speicherung der Informationen in Log-Dateien, ist eine Implementierung in ein SIEM nur über Umwege möglich. In diesem Fall müssten die Dateien von einer Software eingelesen und normalisiert werden. Da dadurch zusätzliche Aufwände generiert werden, aber nur wenig Mehrwert retourniert wird, ist die Funktionalität nicht für die weitere Implementierung geeignet.

2.7.2. System Monitor (sysmon)

Sysmon ist ein Windows-Systemservice und Gerätetreiber, welcher auf Windows-Systemen installiert werden kann. Dieser Systemdienst protokolliert zusätzliche Informationen und Ereignisse in den Windows-Eventlog. Die protokollierten Daten beinhalten Details zu Prozesserstellungen (inkl. Hashwert der ausführbaren Datei), Netzwerkverbindungen und Änderungen an Daten, Registry-Schlüsseln und Systemeinstellungen [28]. Die Hashwerte von ausgeführten Programmen und Netzwerkverbindungen des Systems werden

für eine Analyse mithilfe der Open-Source-Threat-Intelligence-Informationen benötigt. Für eine vollständige Auswertung mithilfe der Windows-Sigma-Regeln sind die Informationen des Sysmon-Service unbedingt erforderlich. Die Tabelle 2.5 zeigt die möglichen Windows-Events, welche durch Sysmon generiert werden können.

Event ID	**Name**
1	Process creation
2	A process changed a file creation time
3	Network connection
4	Sysmon service state changed
5	Process terminated
6	Driver loaded
7	Image loaded
8	CreateRemoteThread
9	RawAccessRead
10	ProcessAccess
11	FileCreate
12	RegistryEvent (Object create and delete)
12	REGISTRYEVENT (OBJECT CREATE AND DELETE)
13	RegistryEvent (Value Set)
14	RegistryEvent (Key and Value Rename)
15	FileCreateStreamHash
16	ServiceConfigurationChange
17	PipeEvent (Pipe Created)
18	PipeEvent (Pipe Connected)
19	WmiEvent (WmiEventFilter activity detected)
20	WmiEvent (WmiEventConsumer activity detected)
21	WmiEvent (WmiEventConsumerToFilter activity detected)
22	DNSEvent (DNS query)
23	FileDelete (A file delete was detected)
24	ClipboardChange (New content in the clipboard)
25	ProcessTampering (Process image change)

255	Error

Tabelle 2.5.: Sysmon Event IDs [29]

Eine Liste der verfügbaren Event-IDs und deren Beschreibung befindet sich im Anhang A.3

Der Sysmon-Dienst muss für die Protokollierung auf den gewünschten Windows-Systemen installiert werden. Das Logging-Verhalten des Service wird über eine Konfigurationsdatei gesteuert. In dieser Datei können einzelne Event-IDs aktiviert, deaktiviert oder bestimmte Regeln für die Log-Generierung angegeben werden. Die Konfigurationseinstellungen sollten für die jeweilige Installationsumgebung unbedingt angewandt werden. Da der Sysmon-Dienst eine große Menge an zusätzlichen Informationen protokolliert, besteht die Gefahr, dass das Log-Volumen ohne spezielle Konfiguration nicht verwendbar wäre.
Ein Beispiel für eine Regel in der Sysmon-Konfiguration ist eine Ausnahme für die Protokollierung der Netzwerkverbindungen. Folgende Regel deaktiviert die Protokollierung von Verbindungen, welche über den Internet Explorer durchgeführt werden:

```
<NetworkConnect onmatch="exclude"> <Image name="network iexplore
   " condition="contains">iexplore.exe</Image> </NetworkConnect>
```

Listing 2.16: Sysmon Ausnahmeregel

Da viele protokollierte Ereignisse durch Standardfunktionen des Betriebssystems oder Basisfunktionen generiert werden, besteht die Notwendigkeit eine Vielzahl von Ausnahmeregeln zu implementieren und somit das Log-Volumen zu senken. Für die Basis einer Konfigurationsdatei bietet es sich an, eine frei verfügbare Konfigurationsdatei, welche die wesentlichen Bestandteile bereits aufweist, zu verwenden. Ein Beispiel einer solchen Konfigurationsdatei ist die „SwiftOnSecurity Sysmon“ Konfiguration.[30] Bei dieser Konfigurationsdatei wurde darauf geachtet, dass Log-Volumen auf wesentliche Ereignisse einzuschränken. Die Regeln der Datei sind so aufgebaut, dass diese als Basiskonfiguration in beinahe jeder Umgebung verwendet werden können.
Da ein Sysmon Regelwerk durch die speziellen Regeln sehr umfangreich und unübersichtlich ausfallen kann („SwiftOnSecurity“ Konfiguration umfasst 1073 Zeilen), ist eine Anpassung des Regelwerks umständlich. Auf Basis der „SwiftOnSecurity“ Sysmon-Konfiguration wurde das Projekt „Sysmon-Modular“ [31] gestartet. Dieses Projekt versucht die Unübersichtlichkeit einer einzelnen langen Konfigurationsdatei mit Modulen zu lösen. Es werden die unterschiedlichen Log-Kategorien unterteilt und für die einzelnen Regeln eigenständige XML-Dateien erstellt. Ein Beispiel hierfür ist eine Ausnahme-Regel für den Prozess

„Windows Defender“. Diese Regel befindet sich im Modul „10_process_access“ und ist in der Datei „exclude_microsoft_windows_defender.xml“ beschrieben. Da der Sysmon-Dienst nur eine Konfigurationsdatei verwenden kann, müssen die gewünschten Regeln vorab zusammengeführt werden. Für diesen Schritt bietet das Projekt ein PowerShell-Skript an. Das Listing 2.17 zeigt, wie das Projekt mithilfe von GIT-Clone heruntergeladen und die Konfigurationsdatei aus allen Regeln erstellt wird.

```
$> git clone https://github.com/olafhartong/sysmon-modular.git
$> cd sysmon modular
$> . .\Merge-SysmonXml.ps1
$> Merge-AllSysmonXml -Path ( Get-ChildItem '[0-9]*\*.xml') -
   AsString | Out-File sysmonconfig.xml
```

Listing 2.17: Sysmon-Modular - Erstellung der Konfigurationsdatei

Für Anpassungen des Regelwerks werden die einzelnen Dateien in den Ordnern erstellt, editiert oder gelöscht. Danach kann die Konfigurationsdatei zusammengeführt und für die Installation des Sysmon-Service verwendet werden. Bei der Installation des Service wird die erstellte Konfigurationsdatei angegeben. Der hierfür notwendige Befehl ist im Listing 2.18 ersichtlich.

```
sysmon -accepteula -i sysmonconfig.xml
```

Listing 2.18: Sysmon - Installation

Bei einem System mit bereits installiertem Sysmon-Service kann bei Änderungen in der Konfigurationsdatei das Regelwerk wie folgt neu geladen werden. Das Listing 2.19 zeigt den hierfür notwendigen Befehl.

```
sysmon.exe -c sysmonconfig.xml
```

Listing 2.19: Sysmon - Aktualisierung der Konfiguration

3. Auswahl der Software-Komponenten

In diesem Kapitel werden die Erkenntnisse der theoretischen Analyse verwendet, um die Auswahl der einzelnen Software-Komponenten und Technologien zu begründen.

3.1. ELK-Stack

Für die Installation wurde das Projekt Open Distro for Elasticsearch (ODFE) gewählt. Bei diesem Projekt handelt es sich um eine Implementierung des ELK-Stacks, welches mit zusätzlichen Funktionen erweitert wurde. Aufgrund der enthaltenen Funktionen in diesem Projekt kann ein Großteil der Anforderungen ohne zusätzliche Lizenzkosten abgebildet werden. Einige dieser Funktionen wären bei der Verwendung des ELK-Stack (Elastic.co) lizenzpflichtig und würden somit zusätzliche Kosten erzeugen. Die Tabelle 3.1 zeigt einen Vergleich der enthaltenen Funktionen zwischen den ELK-Stack-Lizenzen Open-Source, Basic und Open Distro for Elasticsearch.

	ELK Stack (Open Source)	**ELK Stack (Basic)**	**Open Distro for Elasticsearch**
License	Apache 2.0	Elastic Basic	Apache 2.0
Free	Yes	Yes	Yes
Alerting	No	No	Yes
Authentication	No	No	Yes
Access Control	No	No	Yes
Node to Node encryption	No	Yes	Yes
LDAP, AD, Kerberos, SAML Integrations	No	No	Yes
Machine Learning	No	Yes	No
Canvas	No	Yes	No
SQL	No	Yes	Yes
JDBC	No	No	Yes
Elasticsearch Monitoring	No	Yes	Yes
Support	Documentation Forums	Documentation Forums	Documentation Forums
Index-Lifecycle-Management	No	Yes	Yes

Tabelle 3.1.: Funktionsumfang ELK Lizenzen

Einige Funktionen des lizenzpflichtigen X-Pack könnten ebenfalls durch weitere Open-Source-Projekte umgesetzt werden. Ein möglicher Nachteil dieser einzelnen Lösungen könnte die Interoperabilität sein. Werden unterschiedliche Komponenten verwendet, ist es möglich, dass bestimmte Funktionen gewisse Softwareversionen der Komponenten voraussetzen. Bei einem Update des Systems muss der/die Administrator/in diese Überprüfung durchführen und sicherstellen, dass alle Komponenten weiterhin funktionieren. Da ein Großteil der Komponenten in ODFE enthalten ist und man davon ausgehen kann, dass die enthaltenen Pakete untereinander kompatibel sind, entfällt diese Überprüfung durch den/die Administrator/in. Zusätzlich wird das Projekt ODFE in diesem Funktionsumfang auch in weiteren Umgebungen eingesetzt, weswegen mögliche Fehler oft sehr schnell dokumentiert und gelöst werden. Weitere Nachteile der einzelnen Open-Source-Lösungen sind die Weiterentwicklung und Wartung des Projekts. Oft werden diese Software-Projekte von einzelnen Personen gestartet, dadurch werden Anpassungen, Erweiterungen oder Problemlösungen oft auch nur von einer kleinen Gruppe an Entwickler/innen implementiert. Leider kann dies auch

dazu führen, dass keine Weiterentwicklung der Projekte stattfindet. Da das Projekt ODFE durch Amazon-Web-Services (AWS) gestartet und sehr stark weiterentwickelt wird, kann davon ausgegangen werden, dass dieses Projekt noch längere Zeit betreut und gewartet wird. Ein weiterer Faktor, der für die Weiterentwicklung spricht ist, dass dieses Projekt auch bei AWS eingesetzt bzw. auch für Kunden angeboten wird. Dadurch besteht für AWS das Interesse den Funktionsumfang entsprechend zu erweitern und Fehler zu beheben.

3.1.1. Anforderung und mögliche Funktionen

Die definierten Anforderungen an das System werden in folgendem Abschnitt genauer analysiert.

Anreicherung

Die Anforderungen der Anreicherung kann durch die Funktionen der Logstash-Instanz umgesetzt werden. Die Anreicherung wird nach dem Empfangen der Log-Zeilen mit der Pipeline-Konfiguration im Logstash durchgeführt. Da im Open Distro for Elasticsearch keine spezielle Logstash-Version enthalten ist, wird hier der offizielle Logstash-Docker-Container verwendet. Dadurch können auch Funktionen, welche für Logstash bereits vorhanden sind, direkt verwendet werden.

Skalierbarkeit

Die Skalierbarkeit der Installation kann durch zusätzliche Elasticsearch-Instanzen wie auch zusätzliche Kibana- oder Logstash-Instanzen erweitert werden. Da die Funktionsweise der einzelnen Komponenten in diesem Bereich nicht von der Standardinstallation eines ELK-Stacks abweicht, können die Vorteile des ELK-Stacks auch hier angewandt werden.

Lizenzkosten und Verwendungszweck

Open Distro for Elasticsearch basiert auf der Apache 2.0 lizenzierten Version von Kibana und Elasticsearch. Für ODFE wird ebenfalls die Apache 2.0 Lizenzierung angewandt. Dadurch kann die Software ohne Einschränkungen auch bei kommerziellen Verwendungszwecken eingesetzt werden. Die Apache 2.0 Lizenzierung ermöglicht es auch, dass der Source-Code verändert werden kann und nicht veröffentlicht werden muss. Durch die ODFE-Implementierung entstehen keine Lizenzkosten. Die notwendigen Funktionen sind bereits in diesem Projekt integriert, weswegen auch keine weiteren Softwarekomponenten für die Implementierung der Anforderungen notwendig sind.

Mandantenfähigkeit

In Open Distro for Elasticsearch wurde ein mandantenfähiges System integriert. Diese Implementierung bietet durch die Kombination von User, Rollen und Mandanten mehrere Einschränkungsmöglichkeiten. Der Mandant (Tenant) beinhaltet die System-Indizes, welche unter anderem die Kibana-Indizes für die Dashboards und Visualisierungen beinhaltet. Die Rolle definiert, auf welche Indizes sie Zugriff erhalten soll und welche Rechte auf die einzelnen Bereiche erteilt werden. Zusätzlich wird in der Rolle definiert, welche Mandanten mit welchen Rechten der Rolle ersichtlich sein sollen. Die Benutzerkonten bekommen eine Rolle zugewiesen, wodurch die Rechte auf diese vererbt werden. Die Abbildung 3.1 zeigt die Zusammenhänge der einzelnen Elemente.

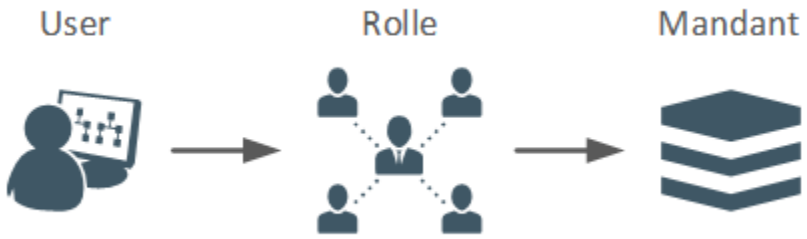

Abbildung 3.1.: Zusammenhang User, Rolle und Mandant

Konfigurationsmöglichkeit einer Rolle

Innerhalb einer Rolle können folgende Rechte konfiguriert werden:

- Index-Permission
- Tenant-Permission
- Cluster-Wide-Permission

Index-Permission

In der Index-Permission können durch die Konfiguration der „index_patterns“ die gewünschten Indizes ausgewählt werden und mit den „allowed_actions“ die Berechtigungen eingeschränkt werden. Zusätzlich bietet diese Funktion auch die Möglichkeit, Berechtigungen auf einzelne Einträge einzuschränken (dsl), einzelne Felder ein- bzw. auszublenden (fls) oder definierte Felder zu anonymisieren („masked_fields“).
Eine Rolle kann mehrere Index-Permissions beinhalten und ermöglicht es dadurch, granulare Rechte zu vergeben. Das Beispiel im Listing 3.1 zeigt die Konfiguration einer Index-Permission mit ausschließlich Leserechten auf alle Indizes welche mit „customer_“ starten und zusätzlich auf die Kibana-System-Indizes für diesen Mandanten.

```
"index_permissions": [
{
  "index_patterns": [
  "customer_*",
  ".kibana*"
  ],
  "dls": "",
  "fls": [],
  "masked_fields": [],
  "allowed_actions": [
  "read"
  ]
}
```

Listing 3.1: ODES Index Permission

Tenant-Permission

In einer „tenant_permission“ können für die einzelnen Mandanten die erlaubten Funktionen angegeben werden. Das Listing 3.2 beinhaltet eine mögliche Konfiguration der Rechte.

```
"tenant_permissions": [
{
  "tenant_patterns": [
  "Customer"
  ],
  "allowed_actions": [
  "kibana_all_read",
  "kibana_all_write"
  ]
}
],
```

Listing 3.2: ODES Tenant Permission

Cluster-Wide-Permission

Diese Berechtigungskonfiguration kann verwendet werden, wenn einer Rolle bestimmte Rechte auf dem

gesamten Cluster zugewiesen werden sollen. Das Beispiel im Listing 3.3 zeigt die Berechtigung für einen Lesezugriff auf alle Daten im Elasticsearch-Cluster.

```
"cluster_permissions": [
"cluster_composite_ops_ro"
]
```

Listing 3.3: ODES Cluster Permission

Durch diese Funktionen wird eine saubere Trennung der Daten für mehrere Mandanten möglich.
Ein offenes Problem bei der Implementierung der Mandantenlösung ist, dass die Monitore und die dazugehörigen Alarme nicht auf Mandanten getrennt sind. Dadurch können die Alarme nur zentral von einem globalen Administrator konfiguriert und verwaltet werden.
Eine mögliche Lösung dieses Problems ist, dass ein ausgelöster Alarm durch die definierten Aktionen in einen eigenen mandantenspezifischen Index geschrieben wird. Durch diese Vorgehensweise können die offenen Alarme auch auf Mandantenebene präsentiert werden.

Alarmierung

Open Distro for Elasticsearch beinhaltet eine eigene Alarmierungsfunktion. Die Alarmierung wird mithilfe von Monitoren angelegt und kann mit den Komponenten Schedule, Input, und Trigger konfiguriert werden. Es werden unterschiedliche Typen von Monitoren angeboten, welche je nach Anforderung ausgewählt werden können. Die zur Auswahl stehenden Typen beinhalten die Definition von Monitoren über „Visual Graph“ , „Elasticsearch query DSL“ oder „Anomaly Detector“.

Visual Graph

Mit diesem Typen können über eine gewünschte Zeitspanne Aggregationen (z.B. count() oder avg()) durchgeführt werden.

Elasticsearch query DSL

Mithilfe des Elasticsearch-Query-Type Domain-Specific-Language (DSL) können direkt DSL-Abfragen in den Monitoren hinterlegt werden. Diese Abfragen erlauben es, dass die Felder der Log-Einträge auf definierte Werte überprüft werden. Hierfür werden die DSL-Abfragen im Monitor hinterlegt und im weiteren Schritt das Ergebnis kontrolliert.
Je nach gewünschtem Monitor kann das Ergebnis der Abfrage nur die Anzahl der Funde oder auch Details aus dem Log-Eintrag beinhalten. Im Fall der Sigma-Regeln können die Monitore auf die Anzahl der Funde

angewandt werden. Der Grund hierfür ist, dass die Regeln immer ein bestimmtes Muster suchen, welches ein Anzeichen für einen Angriff oder Schadcode sein kann. Wird so ein Muster gefunden, soll dieses alarmiert werden.

Anomaly Detector

Dieser Typ von Monitor benötigt einen definierten Anomaly-Detector. Mithilfe diesen Detector kann konfiguriert werden, wie oft und welche Logs überprüft werden sollen. Weicht der jeweilige Durchlauf des Detectors zu stark von den vorhergehenden Durchlaufen ab, kann dies über den Monitor alarmiert werden. Der Schedule der Monitore definiert, wann der Monitor ausgeführt werden soll. Hier werden folgende Funktionen angeboten:

- „By Interval“
  Durch diese Einstellung kann ein Monitor z.B. in regelmäßigen Abständen ausgeführt werden. In der weiteren Implementierung wurde diese Funktion mit einem Wert von fünf Minuten verwendet. Dadurch werden die Monitore alle fünf Minuten ausgeführt und überprüfen die Log-Daten auf die konfigurierten Muster.

- „Daily, Weekly, Monthly“
  Mit diesen Einstellungen kann eine tägliche, wöchentliche oder monatliche Ausführung geplant werden. Bei Auswahl dieser Varianten werden weitere Einstellungsmöglichkeiten angeboten, durch welche die Ausführung zu einer bestimmten Uhrzeit und Wochentag geplant werden kann.

- „Custom cron expression“
  Durch die Verwendung von „custom cron expression“ können die Alarme auch nach dem Crontab-Syntax geplant werden.

Trigger

Die Trigger-Konfiguration bei einem Monitor wird für die weitere Alarmierung verwendet. Ein definierter Trigger kann je nach Monitor-Typ unterschiedliche Werte akzeptieren. Bei der Verwendung von Elasticseach-query-DSL können im Trigger die Ergebnisse der DSL-Abfrage verarbeitet werden. Bei der Trigger Konfiguration werden Überprüfungen der retournierten Abfragewerte durchgeführt. Liefert diese Überprüfung ein positives Ergebnis (return true) wird der Alarm ausgelöst.
In der Trigger Konfiguration stehen mehrere Variablen für die Definition zur Verfügung. Eine Beschreibung dieser befindet sich in der Tabelle 3.2

Variable	Beschreibung
ctx.result	Array mit dem Resultat der DSL-Abfrage
ctx.monitor	Informationen über den konfigurierten Monitor
ctx.trigger	Informationen über den konfigurierten Trigger
ctx.periodStart	Eine Periode wird durch das Interval des konfigurierten Monitors definiert. Wenn ein Alarm alle 5 Minuten ausgeführt wird, ist die „ctx.periodStart“ die aktuelle Zeit der Ausführung weniger fünf Minuten.
ctx.periodEnd	Wie bei „ctx.periodStart“ abhängig von dem Intervall. Bei dem oben genannten Beispiel wäre die „ctx.periodEnd“ der Zeitpunkt des abgeschossen Durchlaufs des Monitor
ctx.error	Beinhaltet Error Meldungen, wenn kein Ergebnis retourniert wurde oder es ein Problem bei der Auswertung der Ergebnisse gab.
ctx.alert	Informationen über den aktiven Alarm

Tabelle 3.2.: Trigger-Variablenbeschreibung

Ein konkretes Konfigurationsbeispiel für einen Trigger, welcher bei einem Fund von nur einem Eintrag alarmieren soll, befindet sich im Listing 3.4.

```
ctx.results[0].hits.total.value > 0
```

Listing 3.4: Trigger bei nur einem Fund

Die „ctx“ Variable beinhaltet die retournierten Werte der DSL-Abfrage und kann entsprechend ausgewertet werden. So können z.B. auch Aggregationen innerhalb des Triggers angewandt werden und bei den definierten Grenzwerten alarmieren.
Der Trigger im Listing 3.5 zeigt eine Durchschnittsberechnung der CPU-Auslastung und alarmiert, wenn der Durchschnitt 90 überschreitet.

```
if (ctx.results[0].aggregations.avg_cpu.value > 90) {
  return true;
}
```

Listing 3.5: Trigger mit Durchschnittsfunktion

Action (optional)

In einem Monitor können zusätzlich Aktionen definiert werden. Diese beinhalten die Aktion, welche durchgeführt werden soll, wenn ein Trigger ein positives Ergebnis liefert. Zur Auswahl stehen Mail-Benachrichtigung, Slack, Amazon Chime oder auch Webhooks. Mithilfe des Webhooks ist es möglich, über den Trigger eine API eines weiteren Systems anzusprechen. Für die Nachricht einer Aktion können ebenfalls die „ctx“ Variablen, welche für den Trigger zur Verfügung stehen, verwendet werden.

3.2. Sigma-Integrationsmöglichkeiten

Da es sich bei dem Sigma-Regelformat um eine generische Beschreibung der Regeln handelt, kann dieses nicht direkt für Suchabfragen verwendet werden. In dem definierten Format der Regeln sind die wesentlichen Parameter enthalten, um aus diesen Informationen entsprechende Abfragen für unterschiedliche Zielsysteme zu erstellen. Das Sigma-Projekt bietet hierfür den Konverter „sigmac“ an, welcher zurzeit eine Umwandlung der Regeln auf folgende Zielsysteme unterstützt:

- Splunk (plainqueries and dashboards)
- ElasticSearch Query Strings
- ElasticSearch Query DSL
- Kibana
- Elastic X-Pack Watcher
- Logpoint
- Microsoft Defender Advanced Threat Protection (MDATP)
- Azure Sentinel / Azure Log Analytics
- Sumologic
- ArcSight
- QRadar
- Qualys
- RSA NetWitness
- PowerShell

- Grep with Perl-compatible regular expression support
- LimaCharlie
- ee-outliers
- Structured Threat Information Expression (STIX)
- uberAgent ESA

Damit die Regeln korrekt auf die gespeicherten Daten angewandt werden können, muss beim Konvertieren eine entsprechende Konfigurationsdatei angegeben werden, welche die zu verwendenden Speicherorte der Logs und die Feldnamen beinhaltet. Das Programm „sigmac“ verwendet diese Konfigurationsdatei und erstellt automatisch die Abfrage für das gewünschte Zielsystem.
Im Fall des ELK-Stacks muss die jeweilige Logquelle auf den definierten Index im Elasticsearch übersetzt werden. Hierfür wird das in der Sigma-Regel definierte Feld „logsource“ verwendet und der gewünschte Index angegeben. Das Listing 3.6 zeigt die Zuweisung des „logstash-windows-* “ Wildcard-Index-Muster.

```
logsources:
windows:
product: windows
index: logstash-windows-*
```

Listing 3.6: Sigmac Logquelle logstash

Sind in der Konfigurationsdatei nicht alle Logquellen zugewiesen, kann mithilfe des „defaultindex“ Parameter ein Standardindex angegeben werden, welcher in diesem Fall verwendet werden soll. Im Listing 3.7 wird der Default-Index auf „logstash-* “ konfiguriert.

```
defaultindex: logstash-*
```

Listing 3.7: Sigmac Standardwert für Logquelle

Damit die Feldnamen aus der Sigma-Regel auf die korrekten Felder der gespeicherten Logeinträge angewandt werden, kann mithilfe des Parameters „fieldmappings“ die Zuweisung vorgenommen werden.
Das Listing 3.8 zeigt die Zuweisung von den Sigma-Regel-Felder EventID, AccessMask, AccountName auf die dazugehörigen Elasticsearch-Document-Felder:

```
fieldmappings:
EventID: winlog.event_id
AccessMask: winlog.event_data.AccessMask
```

```
4  AccountName: winlog.event_data.AccountName
```

Listing 3.8: Sigmac Zuweisung der Feldnamen

3.2.1. Quellen

Eine Sammlung von öffentlichen Sigma-Regeldefinition wird im offiziellen Github-Projekt gepflegt (Neo23x0/sigma). Dieses Projekt beinhaltet zurzeit 694 unterschiedliche Regeln, welche strukturiert abgespeichert und auf folgende Kategorien aufgeteilt sind:

- application
- apt
- cloud
- compliance
- generic
- linux
- network
- proxy
- web
- windows

Dadurch können je nach Anwendungsfall nur die notwendigen Kategorien verwendet werden. Durch die angebotenen GIT-Funktionen können Änderungen am Regelwerk synchronisiert und Änderungen am Regelwerk über die Änderungshistorie nachvollzogen werden.

3.3. Cyber-Threat-Intelligence-Plattformen

Aufgrund der Menge an verschiedenen IOCs und eine stetig steigende Verfügbarkeit von weiteren Quellen ist es notwendig, dass diese zentral verwaltet und gesteuert werden können. Hierfür ist es auch erforderlich, dass die zentrale Verwaltung der IOCs mit zukünftigen Quellen erweitert werden kann. Nachfolgend werden die Lösungen erläutert, welche für die Arbeit in Betracht gezogen wurden.

3.3.1. Alien Labs Open Threat Exchange (OTX)

Das Unternehmen AlienVault bietet für Open-Source-Threat-Intelligence die Plattform Open-Threat-Exchange (OTX) an. Diese Plattform ermöglicht es der Community, Indicator-of-Compromise über eine Programmierschnittstelle oder das Web-Interface auszutauschen. Innerhalb der Alien-Labs-OTX-Plattform werden die unterschiedlichen Indicator-of-Compromise in sogenannten Pulse zusammengefasst. Diese Pulse beinhaltet eine Zusammenfassung und die dazugehörigen IOCs.
Bei OTX handelt es sich um eine reine Online-Lösung, welche über die API in weitere Produkte integriert werden kann. Aufgrund der Tatsache, dass diese Lösung als reine Cloud-Lösung angeboten wird und die Pulse automatisch geteilt werden, besteht keine Möglichkeit, eigene IOCs ausschließlich für den internen Gebrauch zu erstellen. Dies führt dazu, dass die Flexibilität von OTX für eine direkte Weiterverwendung nicht gegeben ist.

3.3.2. Malware Information Sharing Platform and Threat Sharing (MISP)

Bei MISP handelt es sich um eine Open-Source-Threat-Intelligence-Plattform, welche es ermöglicht IOCs auszutauschen. Dieses Produkt kann einerseits als Online-Service direkt bezogen oder auch als On-Premise-Installation betrieben werden. Das Projekt wird durch die Europäische Union finanziell unterstützt und aktiv von mehreren CERTs verwendet (z.B. CERT-EU, CIRCL). Die MISP-Plattform bietet die Möglichkeit von mehreren Quellen Informationen zu beziehen und für die Weitergabe vorzubereiten. So ermöglicht es die Plattform, eigene Events zu erstellen und zu definieren, mit wen diese geteilt werden sollen. Dadurch können Events und die dazugehörigen IOCs nur für den internen Gebrauch definiert werden. Die Events können ebenfalls von unterschiedlichen Online-Services zur Verfügung gestellt werden. So können direkt die offiziellen MISP-Events heruntergeladen und verwendet werden. Zusätzlich können aber auch direkt CSV- oder Text-Listen importiert werden, welche z.B. einen einfachen Import von schadhaften IP-Adressen oder Domainen in einer Listenform ermöglicht.
Die Vorbereitung für die Weiterverwendung wird mit den sogenannten Tags durchgeführt. Die erstellten oder importierten Events können mit Tags versehen werden. Diese Tags können wiederum bei einem späteren Abruf über die API oder das Webinterface gefiltert werden. Eine weitere Filterfunktion der MISP-Plattform wird mithilfe der Organisation bereitgestellt. Damit Events exportierbar sind, müssen diese zuerst veröffentlicht werden. Bei der Veröffentlichung wird definiert, für welche Organisation dieses Event zur Verfügung stehen soll.
Durch den großen Funktionsumfang der Plattform und der integrierten API stehen viele Möglichkeiten zur Verfügung, wie Informationen in weiteren Systemen abgerufen und verarbeitet werden können. Aufgrund dieser Flexibilität wurde die MISP-Plattform für die weitere Verwendung gewählt.

4. Implementierung

Da die Applikationen des Tactical-SIEM innerhalb von Docker-Containern betrieben werden, wurden fünf virtuelle Maschinen als Basis für den Container-Betrieb definiert.
Folgende virtuelle Server wurde mit dem Betriebssystem Ubuntu 18.04 LTS installiert:

- Ein Server für den Logstash Container
- Drei Server mit jeweils einem Elasticsearch Container
- Ein Server, welcher den Kibana und nginx Container bereitstellt

Folgende Abbildung (4.1) zeigt die Aufteilung der Docker-Container auf die definierten virtuellen Server.

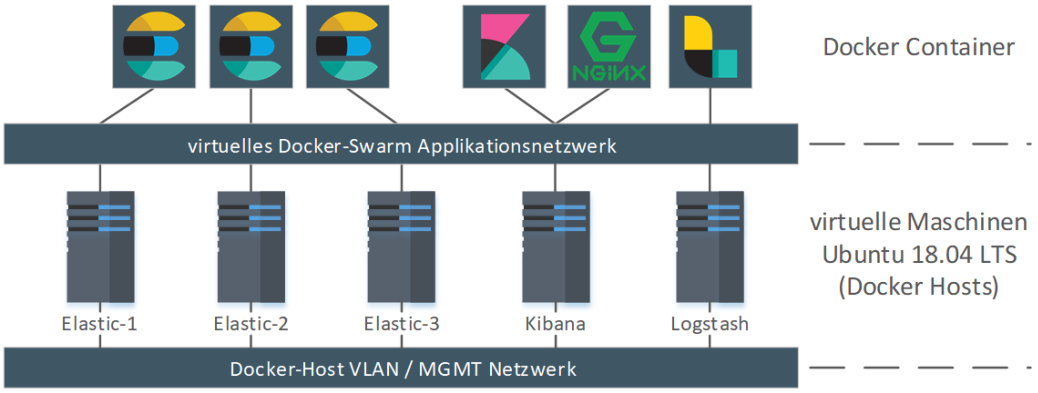

Abbildung 4.1.: Überblick Docker-Container

Durch die Verwendung von Docker können die einzelnen Container auch zu einem späteren Zeitpunkt zwischen den einzelnen Servern verschoben oder auf zusätzliche Server übersiedelt werden. Es besteht ebenfalls die Möglichkeit, die Umgebung mit zusätzlichen Docker-Containern zu erweitern und somit die Performance, den Speicherplatz oder die Redundanzen zu erhöhen.

Die Installation und Konfiguration der einzelnen Komponenten des Tactical-SIEM werden über einen zentralen Ansible-Server durchgeführt. Ansible ist ein Open-Source-Automatisierungs-Werkzeug. Mithilfe von

im YAML-Format definierten Systembeschreibungen werden von Ansible die gewünschten Aufgaben automatisiert und wiederholbar ausgeführt. Durch diese Vorgehensweise ist es möglich, dass die Installation in mehreren Varianten umgesetzt wird. Zusätzlich können die Systeme ohne größeren Aufwand erweitert werden.

4.1. Installation der Basis-Infrastruktur

In einem Ansible-Playbook wird definiert, welche Aufgaben auf welchen Systemen ausgeführt werden sollen. Die definierten Aufgaben können entweder direkt verwendet werden (Module) oder es wird im Playbook eine weitere Rolle eingebunden. Die Rolle beinhaltet nun entweder die gewünschten Modulaufrufe oder inkludiert spezielle Tasks, welche wiederum weitere Modulaufrufe beinhalten können [32]. Diese Kombination aus Playbook, Role und Task ermöglicht es, dass definierte technische Aufgaben wiederverwendbar bleiben und ohne größeren Anpassungen auf weitere Systeme übernommen werden können.
Aufgrund der Komplexität der Umgebung wurde bereits bei der Installation darauf geachtet, dass für die Weiterentwicklungen und Testzwecke eine eigene Testumgebung existiert (Staging-Umgebung). Durch die Kombination von Docker-Container und der Ansible-Installationsvariante ist es möglich, beide Umgebungen kurzfristig mit speziellen Versionen zu installieren. Für die Installation der beiden Umgebungen werden im Ansible eigene Playbooks verwendet. In diesen Playbooks werden die speziellen Variablen für die jeweilige Umgebung definiert und im Hintergrund die gleichen Rollen bzw. Tasks verwendet. Bei der Definition der Rollen und Tasks wurde darauf geachtet, dass diese ausschließlich mit Informationen aus dem Playbook oder mit dynamisch ermittelten Daten arbeiten. Dadurch sind in diesen keine Anpassungen für die unterschiedlichen Umgebungen notwendig. Somit ist sichergestellt, dass die beiden Umgebungen immer mit den gleichen technischen Ansible-Aufgaben installiert / aktualisiert werden und die Test-Umgebung mit der produktiven Umgebung ident ist. Das Listing 4.1 zeigt das Playbook für die Installation der Test-Umgebung.

```
1 ---
2 - hosts: tag_stgdocker
3   become: yes
4   roles:
5   - role: ansible-docker
6     # docker_version: 19.03.5
7     docker_master: IX-STG-ELASTIC-01-0
8     docker_default_bridge_nw: 10.111.190.65/26
```

```
9
10  - role: ansible-docker-alz
11    docker_network_name: STG-ALZ
12    docker_app_nw: 10.111.190.128/26
13    docker_master: IX-STG-ELASTIC-01-0
14    elastic_cluster_name: "stgelastic"
15    opendistro_version: 1.10.1
16    logstash_version: 7.9.1
17    kibana_public_dns_record: stg-alz.anlx.cloud
18    kibana_public_ip: 185.55.161.132
19    docker_elastic_image_tag: "v{{ opendistro_version }}"
20    docker_kibana_image_tag: "v{{ opendistro_version }}"
21    docker_nginx_image_tag: "v{{ opendistro_version }}"
22    docker_logstash_image_tag: "v{{ opendistro_version }}"
```

Listing 4.1: Playbook Installation Test-Umgebung

Erläuterung der verwendeten Ansible-Schlüsselwörter:

- `hosts: tag_stgdocker` - Definiert die Server auf welche das Playbook angewandt wird. Die Informationen werden aus dem Ansible-Inventory abgerufen.
- `become: yes` - Führt das Playbook mit „sudo“ aus.
- `roles` - Mithilfe von „roles“ wird angegeben, dass in den darauffolgenden Bereich weitere Rollen inkludiert werden.
- `role: ansible-docker` - Eine Rolle, welche für die Installation der notwendigen Dienste für Docker und die Initialisierung von Docker-Swarm zuständig ist.

Variablen für die Docker-Installation / Docker-Swarm-Initialisierung:

- `docker_version: 19.03.5` – Falls die Docker-Versionsnummer fixiert werden muss. Im durchgeführten Testbetrieb gab es noch keine Probleme mit der aktuellsten Version, weswegen diese Variable zurzeit nicht verwendet wird.
- `docker_master: IX-STG-ELASTIC-01-0` – Definiert für das Docker-Swarm-Netzwerk den Master-Server (Hostname im Inventory)

- `docker_default_bridge_nw: 10.111.190.65/26` – IP-Bereich für Docker-Swarm-Bridge-Netzwerk

Variablen, welche für die Rolle „ansible-docker-alz“ benötigt werden:

- `docker_network_name: STG-ALZ` – Definiert den Namen des Docker-Swarm-Netzwerks
- `docker_app_nw: 10.111.190.128/26` – IP-Bereich für Docker-Swarm-Applikationsnetzwerk, über welches die Docker-Container untereinander kommunizieren.
- `elastic_cluster_name: "stgelastic"` – Bei der Initialisierung eines Elasticsearch-Cluster muss ein Name definiert werden.
- `opendistro_version: 1.10.1` – Versionsnummer der gewünschten Open-Distro-for-Elasticsearch-Version
- `logstash_version: 7.9.1` – Versionsnummer der gewünschten Logstash-Version
- `kibana_public_dns_record: stg-alz.anlx.cloud` – Hostname für die Kibana-Weboberfläche
- `kibana_public_ip: 185.55.161.132` – IP-Adresse der Kibana-Weboberfläche

Die Versionsnummern wurden im Playbook definiert, damit in der darauffolgenden Rolle spezielle Versionen installiert werden können. Diese Funktionalität wurde integriert, um bei möglichen Problemen oder Tests eine neue Umgebung in einer bestimmten Version installieren zu können. Zusätzlich ermöglicht dies eine Testumgebung mit einer neu veröffentlichten Version, ohne zusätzliche Aufwände zu installieren.
Um für eine neue Software-Version der Komponenten ein neues Docker-Image zu erstellen, wird in der später erläuterten Rolle die Versionsnummer in den Docker-Image-Namen aufgenommen. Der Grund hierfür ist, dass ein neues Docker-Image nur erstellt werden kann, wenn noch kein Image mit dem gewünschten Namen vorhanden ist. Da eine Aktualisierung der Komponenten zurzeit nur gemeinsam mit einer Aktualisierung der ODES-Version sinnvoll ist, wurde die ODES-Versionsnummer für die Ergänzung des Image-Namens verwendet. Damit ein Wechsel der Image Tags, falls dies zu einem späteren Zeitpunkt notwendig wird, einfach umgesetzt werden kann, wurden für die einzelnen Container eigene Variablen definiert:

- `docker_elastic_image_tag: "v{{ opendistro_version }}"`
- `docker_kibana_image_tag: "v{{ opendistro_version }}"`
- `docker_nginx_image_tag: "v{{ opendistro_version }}"`

- `docker_logstash_image_tag: "v{{ opendistro_version }}"`

Bei dem erwähnten Playbook im Listing 4.1 erhalten die jeweiligen „image_tag“ Variablen den Inhalt v1.10.1. Das Listing 4.2 zeigt das Playbook, welches die produktive Umgebung installiert. In diesem Playbook werden die gleichen Variablen wie in der Test-Umgebung, jedoch mit anderen Werten verwendet.

```
---
- hosts: tag_prddocker
become: yes
roles:
- role: ansible-docker
# docker_version: 19.03.5
docker_master: IX-PRD-ELASTIC-01-0
docker_default_bridge_nw: 10.110.190.65/26

- role: ansible-docker-alz
docker_network_name: PRD-ALZ
docker_app_nw: 10.110.190.128/26
docker_master: IX-PRD-ELASTIC-01-0
elastic_cluster_name: "prdelastic"
opendistro_version: 1.10.0
logstash_version: 7.8.3
kibana_public_dns_record: alz.anlx.cloud
kibana_public_ip: 185.55.161.133
docker_elastic_image_tag: "v{{ opendistro_version }}"
docker_kibana_image_tag: "v{{ opendistro_version }}"
docker_nginx_image_tag: "v{{ opendistro_version }}"
docker_logstash_image_tag: "v{{ opendistro_version }}"
```

Listing 4.2: Playbook Installation produktive Umgebung

Durch die angepassten Werte wird die produktive Umgebung zwar mit den gleichen Rollen und Tasks installiert, verwendet aber unterschiedliche Software-Versionen.

Die Rolle „ansible-docker“ installiert auf den ausgewählten Systemen die Dienste: docker-ce, docker und python3-pip. Zusätzlich wird für Elasticsearch empfohlenen die Einstellungen

„vm.max_map_count“ auf mindestens den Wert 262144 zu konfigurieren [33]. Diese Einstellung gibt an, wie viele Arbeitsspeicherbereiche ein einzelner Prozess anfordern kann und ist wie folgt in der Linux Kernel Dokumentation beschrieben :

> „max_map_count:
>
> This file contains the maximum number of memory map areas a process may have. Memory map areas are used as a side-effect of calling malloc, directly by mmap, mprotect, and madvise, and also when loading shared libraries.
>
> While most applications need less than a thousand maps, certain programs, particularly malloc debuggers, may consume lots of them, e.g., up to one or two maps per allocation.
>
> The default value is 65536.“ [34]

Nach einer erfolgreichen Installation der Dienste und Anpassung der Einstellungen, wird der Docker-Swarm initialisiert und die virtuellen Maschinen hinzugefügt. Für diese Schritte ist die zuvor definierte „docker_master“ Variable notwendig. Bei der Initialisierung des Docker-Swarm muss ein Master definiert werden bzw. auf diesem die Initialisierung gestartet werden. Die restlichen Server werden, wie im Listing 4.3 ersichtlich, zu dem neu erstellten Docker-Swarm hinzufügt.

```
1  - name: init_swarm
2  docker_swarm:
3  state: present
4  register: swarm_var
5  delegate_to: "{{ docker_master }}"
6
7  - name: join_swarm
8  docker_swarm:
9  state: join
10 join_token: "{{ swarm_var.swarm_facts.JoinTokens.Manager }}"
11 remote_addrs: [ "{{ docker_master }}:2377" ]
12 advertise_addr: hostvars[inventory_hostname]['
    ansible_default_ipv4']['address']
13 when: inventory_hostname != docker_master
```

Listing 4.3: Docker Swarm Initialisierung

Die Rolle „ansible-docker-alz“ inkludiert die einzelnen Tasks für die Installation und Konfiguration der verschiedenen Applikationen. Die Vorgehensweise für die Erstellung der Docker-Container ist für die einzelnen Container ident. Es wird über Ansible ein bereits vorbereitetes Dockerfile als Template bei der Installation angepasst und für die die Image-Erstellung kopiert. Ein Dockerfile beinhaltet die grundlegenden Informationen für die Image-Erstellung. Das Listing 4.4 zeigt ein Beispiel eines Elasticsearch-Dockerfile:

```
1 FROM amazon/opendistro-for-elasticsearch:{{ opendistro_version
    }}
2 COPY ./docker-entrypoint.sh /usr/local/bin/docker-entrypoint.
   sh
3 COPY ./jvm.options /usr/share/elasticsearch/config/jvm.options
4 LABEL "co.elastic.metrics/module"="elasticsearch" "co.elastic.
   metrics/raw"="[{\"enabled\":true,\"metricsets\":[\"default
   \"],\"module\":\"mockmoduledefaults\",\"period\":\"1m\",\"
   timeout\":\"3s\"}]"
```

Listing 4.4: Dockerfile für Elasticsearch Container

In dem Dockerfile für die Elasticsearch-Container wird angegeben, welches Image als Basis für die Container verwendet werden soll. In diesem Fall wird das von Amazon angebotene Open Distro for Elasticsearch in der gewünschten Version verwendet. Die Version wird durch die im Playbook angegebene „opendistro_version“ Variable definiert.

Zusätzlich werden bei der Container-Erstellung bereits benötigte Dateien in den Container kopiert. Bei diesen Dateien sollte man darauf achten, dass es sich hier um statische Dateien handelt, welche für die Funktionalität essenziell sind. Da diese Dateien nur bei einer Image-Erstellung kopiert werden, würden dies bedeuten, dass bei jeder Änderung in den Dateien ein neues Image erstellt werden muss.

In dem Fall des Elasticsearch-Containers wurden folgende Dateien bei der Image-Erstellung verwendet.

Die „docker-entrypoint.sh“ Datei ist ein Shell-Skript welches für die Erstellung eines neuen Elasticsearch-Clusters ausgeführt werden muss. Die „jvm.options“ beinhaltet Java-Einstellungen für die Anpassungen von z.B. Heap-Speichergrößen oder Encoding-Einstellungen.

Ein Docker-Container kann mithilfe des Ansible-Modul „docker_build “ erstellt werden. Dieses Modul erstellt das Image auf den definierten Servern unter den angegeben Pfaden. Da Ansible die Befehle nur auf dem Zielsystem ausführt und diesen Fall direkt auf dem Zielsystem das Docker-Image erzeugt, muss bereits vorab das korrekte Dockerfile auf das Zielsystem übertragen werden.

Die Ansible Schritte im Listing 4.5 kopieren ein lokales Dockerfile auf die Zielsysteme und erstellen auf diesen das Image:

```
- name: copy_Dockerfile_template
template:
src: "elastic/Dockerfile"
dest: "{{ docker_build_path }}"

- name: docker_build
docker_image:
name: "{{ docker_image_name }}"
build:
path: "{{ docker_build_path }}
pull: yes
tag: "{{ docker_image_tag }}"
source: build
```

Listing 4.5: Ansible Docker-Image Build

Ist es notwendig, dass ein Container einen persistenten Speicherplatz verwendet, kann mithilfe des Ansible-Module „docker_volume“ ein Docker-Volumen erstellt werden. Ohne diesem können die Daten nur innerhalb des Docker-Containers gespeichert werden und würden nach einem Neustart des Containers nicht mehr zur Verfügung stehen.
Das Beispiel im Listing 4.6 zeigt, wie ein Docker-Volumen für die Elasticsearch-Container angelegt wird. Als Name für das Volumen wurde der Hostname der jeweiligen virtuellen Maschine gewählt.

```
- name: create_docker_volume
docker_volume:
name: "{{ inventory_hostname }}"
when: "'tag_elastic' in group_names"
```

Listing 4.6: Ansible Docker Volume

Damit ein Docker-Container mit den gewünschten Einstellungen gestartet wird, kann das Ansible das Modul „docker_container“ verwendet werden. Dieses Modul kann das zuvor erstellte Docker-Image als Docker-Container starten. Hierfür müssen folgende Parameter definiert werden:

- `Name` - Name des Docker-Container
- `Image` - Name des Images, welches zuvor erstellt wurde.

Folgende Parameter sind optional, werden jedoch für die Funktionsweise aller Container der Umgebung benötigt:

- `Networks` – Name: Name des Docker-Netzwerks, über welches der Docker-Container kommunizieren soll.
- `Published Ports` - TCP- / UDP-Ports, welche außerhalb des Docker-Netzwerk erreichbar sein sollen. Die Services auf diesen Ports sind über die IP-Adresse der virtuellen Maschine, welche den Docker-Container betreibt, erreichbar.
- `Volumes` - Mithilfe dieses Parameters wird das zuvor erstellte Docker-Volume in den jeweiligen Container bereitgestellt. Dadurch können im Containern die Daten direkt am Docker-Host abgespeichert werden und stehen nach einem Docker-Container Neustart wieder zur Verfügung.

Das Listing 4.7 zeigt den Aufruf des „docker_container“ Moduls.

```
- name:
docker_container:
name: "{{ inventory_hostname }}"
networks:
- name: "{{ docker_network_name }}"
image: "{{ docker_image_name }}:{{ docker_image_tag }}"
published_ports:
- 9200:9200/tcp
volumes:
- "{{ inventory_hostname }}:/usr/share/elasticsearch/data"
```

Listing 4.7: Ansible Docker-Container

4.2. Konfiguration ELK-Stack

Der ELK-Stack besteht aus den drei Software-Produkten Elasticsearch, Logstash und Kibana. Diese Applikationen werden auf den vorbereiteten virtuellen Maschinen als Docker-Container betrieben. Folgendes Kapitel beschreibt die Konfiguration dieser Produkte.

4.2.1. Elasticsearch

Für die Einstellungen der Elasticsearch-Installation werden folgende Konfigurationsdateien verwendet [35].

- elasticsearch.yml - Konfiguration der Elasticsearch-Node und der Elasticsearch-Cluster Einstellungen
- jvm.options - Konfiguration der Java-VM-Einstellungen
- log4j2.properties - Konfiguration der Elasticsearch-Logging-Einstellungen

Elasticsearch-Konfiguration

Die Elasticsearch-Einstellungen innerhalb der „elasticsearch.yml “ sind aufgeteilt in Node und Cluster spezifische Einstellungen. Bei den Node spezifischen Einstellungen werden die Namen und Funktionen einer einzelnen Elasticsearch-Node konfiguriert. Das Listing (4.8) zeigt die Node-Einstellungen, welche das Verhalten der Nodes im Elasticsearch-Cluster bestimmen.

```
1 node.name: {{ inventory_hostname }}
2 node.master: true
3 node.data: true
4 node.ingest: true
```

Listing 4.8: Elasticsearch Node spezifische Einstellungen

Mithilfe der „node.name“ wird der Name der Elasticsearch-Node auf den Hostnamen im Ansible-Inventory des Servers gesetzt. In einem Elasticsearch-Cluster muss ein sogenannter „Master“ ausgewählt werden, damit dieser die Steuerung des Clusters übernimmt. Ob eine Node diese Funktion übernehmen darf wird durch die Einstellung „node.master“ bestimmt. Die Einstellung „node.data“ definiert, ob eine Elasticsearch-Node für die Datenspeicherung verwendet werden darf. Durch „node.ingeset“ wird definiert, dass die konfigurierte Node-Daten in den Elasticsearch-Cluster hinzufügen kann.
Grundsätzlich können alle Funktionen von der gleichen Elasticsearch-Node übernommen werden. In kleineren Installationen ist dies auch sinnvoll, damit Redundanzen in einem Fehlerfall zur Verfügung stehen. Handelt es sich um eine größere Elasticsearch-Installation, kann in Betracht gezogen werden, dass nur ausgewählte Nodes für die Master-, Data- oder Ingest-Rolle qualifiziert sein sollen. Dadurch kann der Ressourcenbedarf optimiert werden und die Performance des Clusters verbessert werden.
Da die Tactical-SIEM-Installation aus drei Elasticsearch-Nodes besteht, wurden allen Nodes die drei Rollen zugewiesen. Die Auswahl des Master-Nodes wird automatisch durch den Elasticsearch-Cluster vorgenommen. Fällt durch einen Fehlerfall der bestehende Master-Node aus, wird die Rolle automatisch auf den nächsten Node verschoben.

Die Cluster spezifischen Einstellungen definieren den Cluster-Namen und die IP-Adressen, auf welche der Elasticsearch-Dienst gebunden wird („network.host“) bzw. die Adresse, welche im Cluster mit den anderen Nodes ausgetauscht wird („network.publish_host“). Das Listing 4.9 zeigt den Auszug aus der Konfigurationdatei, welche von Ansible verwendet wird.

```
cluster.name: {{ elastic_cluster_name }}
network.host: 0.0.0.0
network.publish_host: {{ inventory_hostname }}
```

Listing 4.9: Elasticsearch Cluster spezifische Einstellunge

Um zu verhindern, dass der Arbeitsspeicher durch den JavaVM-Garbage-Collector bereinigt und auf die Festplatte ausgelagert wird, kann durch die im Listing 4.10 ersichtliche Einstellung der Speicherbereich für den Garbage-Collector gesperrt werden.

```
bootstrap.memory_lock: true
```

Listing 4.10: Elasticsearch Memory-Lock

Damit ein Elasticsearch-Node einem bestehenden Cluster beitreten kann, ist es notwendig, dass dieser eine Adressliste von mindestens einem bestehenden Node zur Verfügung hat („discovery.seed_hosts“). Da die Installation über Ansible durchgeführt wird, sind zum Zeitpunkt der Installation bereits alle teilnehmenden Nodes bekannt. Für die Konfigurationsdatei werden von Ansible alle teilnehmenden Elasticsearch-Nodes ausgelesen und die Konfigurationsdatei dynamisch erstellt. Hierfür wird eine
Foreach-Schleife verwendet, welche von allen, für Elasticsearch gekennzeichneten Servern, den Hostnamen eingefügt. Hierbei muss darauf geachtet werden, dass Ansible nur die Zielserver des im Moment ausführenden Playbooks verwendet („current_hosts“). Andernfalls könnte die Konfigurationsdatei auch Server mit dem „tag_elastic“ miteinbeziehen, welche für eine andere Umgebung gedacht sind. Im Listing 4.11 ist die verwendete Ansible-Funktion ersichtlich.

```
{% set current_hosts = groups['tag_elastic'] | intersect(
   ansible_play_hosts_all|list) %}
discovery.seed_hosts: {% for host in current_hosts | map('
   extract', hostvars, 'inventory_hostname') | list %}
- {{ host }} {# comment - gets all inventory_hostname with the
   tag tag_elastic #}
{%- endfor %}
```

Listing 4.11: Elasticsearch Node-Liste

Nachdem diese Funktion, bei der Installation, durch Ansible ausgeführt wurde, befindet sich die Elasticsearch-Nodes in folgendem Format (Listing: 4.12) in der Konfigurationsdatei:

```
1  discovery.seed_hosts:
2    - Elasticsearch-Host-1
3    - Elasticsearch-Host-2
```

Listing 4.12: Elasticsearch Nodes

Diese Vorgehensweise wurden ebenfalls für die Einstellung „cluster.initial_master_nodes“ verwendet. Diese Einstellung definiert die möglichen Master-Nodes eines Elasticsearch-Clusters. Da in dieser Umgebung alle teilnehmenden Nodes die Master-Rolle übernehmen können, wurde, wie in Listing 4.13 ersichtlich, hier ebenfalls alle Elasticsearch-Nodes angegeben.

```
1 cluster.initial_master_nodes: {% for host in current_hosts | map
    ('extract', hostvars, 'inventory_hostname') | list %}
2 - {{ host }} {# comment - gets all inventory_hostname with the
    tag tag_elastic #}
3 {%- endfor %}
```

Listing 4.13: Elasticsearch Master-Liste

Die Auswahl des zurzeit aktiven Clusters bzw. die Ermittlung einer Master-Node im Fall eines neuen Clusters wird automatisch von Elasticsearch durchgeführt.

Da die Open-Distro-for-Elasticsearch-Applikation eine Transport-Verschlüsselung zwischen den Elasticsearch-Nodes unterstützt, wurden in der „elasticsearch.yml“ Konfigurationsdatei die notwendigen Zertifikate angegeben. Die Zertifikate werden durch Ansible erstellt und bei der Installation auf die Systeme übertragen. Das Listing .4.14 zeigt die verwendeten Parameter.

```
1 opendistro_security.ssl.transport.pemkey_password: {{
    private_key_passphrase }}
2 opendistro_security.ssl.transport.pemkey_filepath: {{
    inventory_hostname }}.pem
3 opendistro_security.ssl.transport.pemcert_filepath: {{
    inventory_hostname }}.crt
4 opendistro_security.ssl.transport.pemtrustedcas_filepath: ca.crt
```

Listing 4.14: Elasticsearch Transport-Verschlüsselung

Eine weitere Funktion der Transport-Verschlüsselung ist die Identifikation der teilnehmenden Elasticsearch-Nodes. Damit innerhalb eines bestehenden Elasticsearch-Clusters nur erlaubte Nodes Anfragen absetzen können, wurde in der Konfigurationsdatei definiert, dass die Nodes ein Zertifikat mit einem bestimmten Distinguished-Name (DN) besitzen müssen (siehe Listing 4.15). Zusätzlich wird überprüft, ob das Zertifikat von der davor angegeben Certificate-Authority ausgestellt wurde.

```
opendistro_security.nodes_dn:
- "CN=*-Docker,O=domain.tld,C=AT"
```

Listing 4.15: Elasticsearch Transport-Zertifikate

Bei einem Admin-Zertifikat handelt es sich um ein Client-Zertifikat, welches für administrative Tätigkeiten mit erweiterten Rechten benötigt wird. Dies umfasst unter anderem Änderungen an der ODES-Security-Plugin-Konfiguration. Mithilfe der Befehle im Listing 4.16 wurde der DN des Admin-Zertifikats fixiert.

```
opendistro_security.authcz.admin_dn:
- "CN=elastic-admin,O=domain.tld,C=AT"
```

Listing 4.16: Elasticsearch Admin-Zertifkat

Elasticsearch-Funktionen werden über die REST-API (http- / https-Schnittstelle) durchgeführt. Wie in Listing 4.17 ersichtlich, wurde in der Konfigurationsdatei die http-SSL-Verschlüsselung aktiviert und die notwendigen Zertifikate definiert, damit die Übertragung zwischen Elasticsearch und weiteren Komponenten verschlüsselt stattfindet.

```
opendistro_security.ssl.http.enabled: true
opendistro_security.ssl.http.pemkey_password: {{
   private_key_passphrase }}
opendistro_security.ssl.http.pemkey_filepath: {{
   inventory_hostname }}.pem
opendistro_security.ssl.http.pemcert_filepath: {{
   inventory_hostname }}.crt
opendistro_security.ssl.http.pemtrustedcas_filepath: root.crt
opendistro_security.allow_unsafe_democertificates: false
```

Listing 4.17: Elasticsearch REST API - TLS-Verschlüsselung

Der gesamte Inhalt der „elasticsearch.yml“ Konfigurationsdatei befindet sich im Anhang A.1.

JavaVM-Konfiguration

Bei Elasticsearch handelt es sich um eine Java Applikation, welche in einer JavaVM betrieben wird. Einige Standardwerte einer JavaVM müssen für eine Elasticsearch-Installation angepasst werden, damit die gewünschte Stabilität und Performance erreicht wird. Eine Möglichkeit für die Anpassungen der Einstellungen ist die Konfigurationsdatei „jvm.options“. In dieser Datei sind wesentliche Parameter konfiguriert, welche einen direkten Einfluss auf die Performance haben. Ein Beispiel hierfür ist die Auswahl des Java-Garbage-Collector. Bei einer Elasticsearch-Installation wird diese Konfigurationsdatei bereits mitgeliefert. Falls es keine unbedingte Notwendigkeit gibt, sollten Einstellungen wie das Verhalten des Garbage-Collectors nicht angepasst werden. Einstellungsmöglichkeiten wie die Heap-Speichergröße müssen hingegen auf die aktuelle Umgebung adaptiert werden. Die Heap-Speichergröße wird über die Parameter „-Xms“ und „-Xmx“ festgelegt. Der Parameter „-Xmx“ spezifiziert den maximalen Arbeitsspeicherbereich einer Java-VM. Mithilfe des „-Xms“ wird definiert, wie viel Arbeitsspeicher beim Start der virtuellen Maschine bereits initialisiert wird.

Eine weitere Möglichkeit, diese Einstellungen vorzunehmen, ist die Konfiguration der gleichnamigen Umgebungsvariablen. Bei der umgesetzten Elasticsearch-Installation mit Docker-Containern wurde die Konfigurationsvariante über die Umgebungsvariable gewählt. Hierfür wird beim Start des Docker-Container die Umgebungsvariable „ES_JAVA_OPTS“ definiert (Siehe Listing 4.18).

```
1 - name:
2   docker_container:
3     env:
4       ES_JAVA_OPTS: "-Xms6g -Xmx6g"
```

Listing 4.18: Docker Container - Umgebungsvariable

Der Grund für diese Konfigurationsvariante ist, dass die „jvm.options“ Konfigurationsdatei aus dem jeweiligen Elasticsearch-Projekt ohne weitere Anpassungen übernommen werden kann. Die Konfigurationsdatei muss nicht auf die eingesetzten virtuellen Server angepasst werden und ist auf jedem System ident. Da die Installation mithilfe von Ansible durchgeführt wird, ist es auch möglich, dass die „Xmx“ und „Xms“ Werte bei der Installation dynamisch berechnet werden. Ein zusätzlicher Grund für die Konfiguration über die Umgebungsvariable ist, dass bei einer Änderung der Einstellungen nur der Docker-Container mit den neuen Parametern neu gestartet werden muss. Es ist nicht notwendig die Konfigurationsdateien aller Systeme auszutauschen.

Logging-Konfiguration

Die Handhabung der Log-Einträge für den Elasticsearch-Dienst wird über die Konfigurationsdatei „log4j2.properties“ gesteuert. Die Parameter der Konfigurationsdatei bieten Einstellungsmöglichkeiten für die Log-Pfade, Namen der Log-Dateien und die Größe der gespeicherten Log-Dateien. Mithilfe der Log-Rotation kann festgelegt werden, in welchem Abstand neue Log-Dateien erstellt werden (z.B. nach Erreichen einer bestimmten Größe oder nach einem gewissen Zeitraum) und wie lange diese aufbewahrt werden. Zusätzlich kann das Log-Level, welches bestimmt, welche Log-Einträge generiert werden (Informationen, Warnungen oder Fehler), konfiguriert werden. Bei dieser Installation wurden die Standardeinstellungen übernommen [36].

4.2.2. Logstash

Die Logstash-Instanz ist der erste Eintrittspunkt der Log-Daten in die Plattform. Über verschiedene Inputs in der Pipeline-Konfiguration können auf bestimmten Ports unterschiedliche Services empfangen werden. In der Grundinstallation der Plattform werden hauptsächlich Syslog- und Winlogbeat-Nachrichten verwendet. Hierfür wurden, wie in Abbildung 4.2 ersichtlich, die Ports UDP 5514 für Syslog und TCP 5044 für die Logbeat-Kommunikation verwendet.

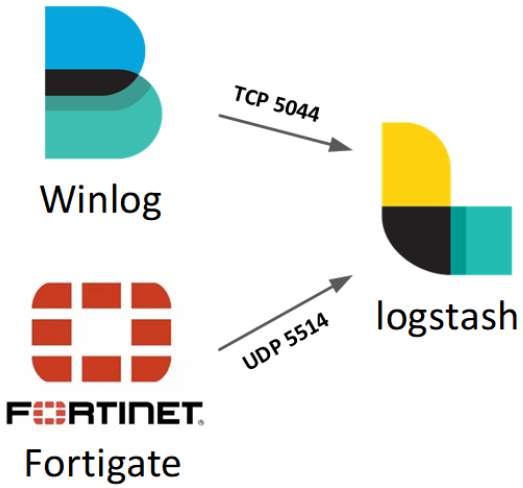

Abbildung 4.2.: Externe Services von Logstash

Index-Pattern

Neben dem Empfangen, Bearbeiten, der Filterung und dem Weiterleiten der Log-Daten wird durch Logstash auch die Typdefinition der einzelnen Felder und weitere Index-Einstellungen für Elasticsearch definiert. Die-

se Konfiguration wird in den „Index pattern“ vorgenommen. Die allgemeinen Einstellungen, welche in den verwendeten Index-Pattern vorgenommen wurden, beinhalten folgende Punkte:

index.refresh_interval

Das Index-Refresh-Interval definiert in welchen Abständen der Elasticsearch-Cluster die Indexierung durchführt. Die Indexierung ist notwendig, damit die Einträge durch Suchabfragen gefunden werden können. Die Standardeinstellung des Index-Refresh-Interval aktualisiert den Index jede Sekunde, aber nur für Indexe, welche innerhalb der letzten 30 Sekunden eine Abfrage erhalten haben.
In diesem Anwendungsfall können regelmäßig automatische Abfragen stattfinden (z.B. Visualisierung und Alarmierung). Dadurch würden im ungünstigsten Fall alle Indexe jede Sekunde aktualisiert werden. Da eine Indexierung entsprechende Ressourcen benötigt und eine sekündliche Aktualisierung nicht erforderlich ist, wurde das Refresh-Interval auf fünf Sekunden fixiert [37].

number_of_shards

Die Anzahl der Shards bestimmt, auf wie viele Shards ein angelegter Index aufgeteilt wird. Durch das Aufteilen des Index können die Elemente auf mehrere Elasticsearch-Nodes aufgeteilt werden. Diese Aufteilung ermöglicht eine Verteilung der Abfragen auf die einzelnen Nodes. Für die Konfiguration der Shard-Anzahl müssen mehrere Parameter beachtet werden. Einerseits ist die Anzahl der Shard abhängig von der Elasticsearch-Node Anzahl und deren Heap-Speicher. So wird z.B. von Amazon AWS empfohlen, maximal 20 Shards pro GiB Java-Heap-Speicher zu planen. Des Weiteren ist die maximale Shard-Anzahl seit der Elasticsearch-Version 7.x auf 1000 Shards pro Nodes limitiert [38]. Bei der Berechnung der Shard-Anzahl muss zusätzlich auf die number_of_replicas geachtet werden. Durch diese Einstellung wird je Shard die konfigurierte Anzahl an Replicas erstellt. Die Shard-Anzahl kann für bestehende Indizes nicht angepasst werden. Bei einer Änderung der Shard-Anzahl müssen die Documents neu indexiert werden.
In diesem Aufbau wurde die Shard Anzahl auf drei fixiert. Der Grund hierfür ist, dass drei Elasticsearch-Nodes in der Plattform vorgesehen sind und der Index täglich pro Gerät bzw. Mandant neu angelegt wird. Dadurch sollten keine Probleme mit einem zu großen Index auftreten. Ein weiterer wesentlicher Faktor ist, dass die Aufbewahrungsdauer für die Indizes gering gewählt wurde (zwischen zehn und 35 Tage) und dadurch die älteren Shards / Indizes schneller entfernt werden. Durch diese relativ geringe Aufbewahrungsdauer und die täglich neu erstellen Indizes, kann auch die Shard-Anzahl ohne größeren Aufwand angepasst werden. Die neu erstellen Indizes werden mit der neuen Shard-Anzahl erstellt. Spätestens nach Ablauf der Aufbewahrungsdauer von zehn bzw. 35 Tage werden die alten Shards entfernt.

number_of_replicas

Durch die number_of_replicas wird angegeben, wie viele Replicas pro Shard erstellt werden. Die Replicas dienen dazu, dass bei einem Ausfall einer Elasticsearch-Node die betroffenen Shards aus den Replicas wiederhergestellt werden können. In der aufgebauten Plattform wurde ein Replica-Anzahl von eins gewählt. Dadurch ist ein Ausfall von einer Elasticsearch-Node ohne Datenverlust möglich.

opendistro.index_state_management.policy_id

Durch die Open Distro for Elasticsearch wird die Funktion Index-State-Management (ISM) zur Verfügung gestellt. Durch diese Funktion kann eine Aufbewahrungsrichtlinie für die erstellen Indizes hinterlegt werden. Diese Aufbewahrungsrichtlinie ermöglicht es, dass die Indizes nach der definierten Aufbewahrungszeit geändert (z.B. gelöscht) werden. Für die Plattform wurden zwei unterschiedliche ISM-Regeln angelegt:

- „delete_index_after_35-days“ – Löscht den Index nach 35 Tagen
- „delete_index_after_10-days“ – Löscht den Index nach zehn Tagen

Das Listing 4.19 zeigt ein Beispiel aus dem Index-Template für Fortigate-Firewalls.

```
"index_patterns" : "*_fortigate_*",
"settings" : {
  "opendistro.index_state_management.policy_id" :
    "delete_index_after_35days",
  "index.refresh_interval" : "5s",
  "number_of_shards" : 3,
  "number_of_replicas" : 1
},
```

Listing 4.19: Index Template für Fortigate Firewall

Field-Mapping

Ein weiterer Bestandteil der Index-Patterns ist das Field-Mapping. In diesem Bereich werden die Datentypen der einzelnen Felder im Index definiert. Diese Zuteilung ist vor allem bei den weiteren Verwendungen in Suchabfragen oder Visualisierungen relevant. Aufgrund des Umfangs der gesamten Typen-Zuteilungen werden hier nur spezielle Felder als Beispiele aufgelistet.

Im Listing 4.20 wird das Feld Timestamp als Type Datum definiert.

```
"@timestamp": {
```

```
  "type": "date"
},
```

Listing 4.20: Zuweisung Datentyp für Timestamp

Durch diese Definition können danach Funktionen für ein Datumfeld verwendet werden.

Als Type IP kann eine IP-Adresse, wie im Listing 4.21 ersichtlich, definiert werden.

```
"srcip": {
  "type": "ip"
},
"dstip": {
  "type": "ip"
},
```

Listing 4.21: Zuweisung Datentyp für IP-Adressen

Diese Definition ist notwendig, damit in den Suchabfragen spezielle Ausdrücke für IP-Adressen verwendet werden können. Dadurch ist es möglich, dass in der Suche z.B. ganze Subnetze oder IP-Bereiche angegeben werden.
Für GEO-Informationen (Koordinaten) kann der Datentype geo_point verwendet werden. Dadurch können die Informationen bei einer Visualisierung auf Karten dargestellt werden. Ein Beispiel hierfür befindet sich im Listing 4.22.

```
"src_geoip.location": {
  "type": "geo_point"
},
"dst_geoip.location": {
  "type": "geo_point"
},
```

Listing 4.22: Zuweisung Datentyp für geographische Informationen

Der Keyword-Datentyp wird benötigt, damit später auf Text-Inhalte der Felder eine Aggregat-Funktionen angewandt werden können. Eine genaue Beschreibung befindet sich im Kapitel 2.3.1. Das Listing 4.23 beinhaltet eine Definition eines Keyword-Feldes.

```
"dynamic_templates": [
{
  "default": {
    "match": "*",
    "mapping": {
      "type": "keyword"
    }
  }
}
```

Listing 4.23: Definition einer dynamischen keyword Zuweisung

Weiteren Feldern wurden der Typ Integer oder Long zugeteilt. Felder welche nicht explizit im Index-Pattern aufgelistet werden, können durch Elasticsearch automatisch erkannt werden.[39] Mögliche Datentypen sind hier: object (JSON), float, double, boolean, text, date

Pipeline-Funktionalität

Folgendes Kapitel beschreibt die unterschiedlichen Funktionen einer Logstash-Pipeline. Als Anwendungsbeispiel wurde hierfür die Syslog-Pipeline verwendet. Im ersten Schritt einer Pipeline wird das Eingabe-Modul, über welches die Daten empfangen werden, definiert. Die Eingabe für die Syslog-Pipeline wurde mit der Konfiguration aus dem Listing 4.24 abgebildet.

```
input {
  udp {
    port => 5514
    tags => "syslog"
  }
}
```

Listing 4.24: Logstash - Input Pipeline Syslog

Durch das Hinzufügen des Tags „syslog“ kann zu späteren Zeitpunkten in der Pipeline oder auch weiteren Pipelines auf diesen Tag gefiltert werden. Im nächsten Abschnitt „filter“ können die übermittelten Logs weiter angepasst werden. Für den Anwendungsfall von Syslog wurde im ersten Schritt überprüft, ob es

sich bei der Syslog-Nachricht um das Format einer Fortigate-Firewall handelt und im gleichen Schritt auch die Feldnamen zugeteilt. Diese Zuteilung ist notwendig, da es sich bei der Syslog-Nachricht um eine unstrukturierte Logzeile handelt. Diese Schritte können kombiniert in der Pipeline-Konfiguration, mithilfe der Index-Pattern und der Funktion grok, durchgeführt werden.
Das im Listing 4.25 ersichtliche „grok“ Muster beschreibt eine normale Fortigate-Syslog-Nachricht und bestimmt die Variablennamen für die ausgelesenen Meta-Informationen. Die Informationen in der Logzeile werden für die weitere Verarbeitung in der Variable „fortigate“ gespeichert. Diese Informationen sind im Key-Value-Format und werden im weiteren Verlauf der Pipeline ausgelesen.

```
patterns_dir => [ "/usr/share/logstash/patterns.d" ]
match => {
  "message" => [ "%{SYSLOG5424PRI:syslog_index}date=%{FORTIDATE:
   date} time=%{TIME:time}
        devname=\"%{HOSTNAME:devname}\" devid=\"%{HOSTNAME:devid
   }\"
        logid=\"%{NUMBER:logid}\" type=\"%{DATA:type}\"
        subtype=\"%{DATA:subtype}\" %{GREEDYDATA:fortigate}" ]
}
add_tag => [ "FortiGate" ]
```

Listing 4.25: Logstash - Syslog Zuweisung der Feldnamen

Wenn die Nachricht erfolgreich ausgelesen werden konnte, wird das Tag „Fortigate“ zu dem Log-Eintrag hinzugefügt. Dies ermöglicht es, dass in der weiteren Pipeline-Konfiguration nun spezielle Fälle für Fortigate-Log-Einträge abgebildet werden können. Das Listing 4.26 zeig ein Beispiel für Fortinet-Fortigate-Syslog-Nachrichten, in welchem der Zeitstempel aus den Logs in das Logstash-Metafeld „@timestamp“ kopiert wird.

```
date {
  match => [ "FORTIDATETIME", "YYYY-MM-dd HH:mm:ss" ]
  timezone => "Europe/Vienna"
  locale => en
  target => "@timestamp"
}
```

Listing 4.26: Logstash - Syslog Timestamp übertragen

Die Key-Value werden, welche zuvor mit der Funktion „grok“ in der Variable „fortigate“ gespeichert wurden, werden mit dem Filter-Plugin „kv“ verarbeitet. Dieses Plugin teilt die Key-Value-Werte auf und im weitere Verlauf der Pipeline können direkt die Feldnamen angesprochen werden. Die verwendete Konfiguration befindet sich im Listing 4.27.

```
kv {
  source => "fortigate"
  field_split => "\s"
  value_split => "="
}
```

Listing 4.27: Logstash - Syslog in Key-Value Werte umwandeln

Das Listing 4.28 zeigt, wie mithilfe der „remove_field“ Funktion die gewünschten Felder entfernt werden können.

```
if "FortiGate" in [tags] {
  mutate {
    remove_field => [ "syslog_index", "year", "month",
    "day", "fortigate", "date", "time", "FORTIDATETIME",
    "message", "syslog5424_pri", "incidentserialno" ]
}
```

Listing 4.28: Logstash - Syslog Entfernung von Felder

Damit die über Syslog übermittelten Fortigate-Log-Einträge auch einen eigenen Mandanten zugeordnet werden können, ist es notwendig, eine eindeutige Information im Log auszuwerten. In diesem Fall wurde die Seriennummer der Fortigate gewählt, da diese im Normalfall durch den Hersteller eindeutig an eine Appliance vergeben wird. Als Basis für diese Zuteilung wurde eine CSV-Datei mit den bekannten Seriennummern und den dazugehörigen Kunden erstellt. Das Listing 4.29 zeigt einen Beispielinhalt der CSV-Datei:

```
"FG200EXXXXXXXXX","customer1"
"FG200EXXXXXXXXX","customer1"
"FGT50EXXXXXXXXX","customer2"
```

Listing 4.29: Logstash - Syslog-Mandanten-Liste

Durch das Logstash-Filter-Plugin „translate“ ist es möglich, Werte aus den Logzeilen mit einer Datei abzugleichen. Somit kann die Seriennummer aus der Logzeile verwendet werden, um eine Zuteilung zu dem

Mandanten durchzuführen. Diese Information kann direkt durch das Plugin in ein neues Feld gespeichert werden. Die umgesetzte Variante ist im Listing 4.30 ersichtlich.

```
translate {
  field => "[devid]"
  destination => "[customtenant]"
  dictionary_path => "/usr/share/logstash/tenants.csv"
  fallback => "unknown"
}
```

Listing 4.30: Logstash - Syslog-Mandanten-Zuweisung

Mithilfe dieser hinzugefügten Information, können bei der Ausgabe der Log-Daten eigene Elasticsearch-Indizes je Mandanten angelegt werden. Wie im Listing 4.31, Zeile 5 ersichtlich, können für die Index-Namen die Feldinhalte verwendet werden.

```
output {
  if "FortiGate" in [tags] {
    elasticsearch {
      hosts => [ Liste der Elasticsearch Hosts ]
        index => "%{customtenant}_fortigate_%{type}-%{+YYYY.MM.
   dd}.log"
        template => "/usr/share/logstash/config/fortigate.json"
        template_name => "fortigate"
        template_overwrite => "true"
        manage_template => "true"
    }
  }
```

Listing 4.31: Logstash - Syslog Output Konfiguration

Winlogbeat-Pipeline

Für die Übermittlung der Windows Event-Logs wurde auf den eingesetzten Windows-Systemen der Winlogbeat-Agent verwendet (nähere Informationen im Kapitel 4.5). Damit diese Informationen von der Logstash-Instanz angenommen, verarbeitet und in den Elasticsearch-Cluster übermittelt werden, wurde eine eigene Logstash-Pipeline-Konfiguration erstellt. Der grundlegende Aufbau dieser Pipeline ähnelt dem Ablauf der Syslog-Pipe.

Wie im Listing 4.32 ersichtlich, wird im ersten Schritt dieser Pipeline konfiguriert, dass Winlogbeat-Agent

Verbindung angenommen werden. Zusätzlich wird pro Mandanten ein eigener Input definiert, weswegen in diesem Schritt das Feld „customtenant“ mit dem Kundennamen hinzugefügt wird.

```
beats {
  port => 5044
  host => "0.0.0.0"
  include_codec_tag => false
  add_field => {
    "customtenant" => "customer1"
  }
}
```

Listing 4.32: Logstash - Pipeline Beat Input

Nachdem die Daten von dem Agenten in der Logstash-Instanz angenommen wurden, werden diese in der Pipeline gefiltert und die Windows-Event-Log-Felder normalisiert. Da die Windows-Event-Logdaten keinen einheitlichen Aufbau besitzen und je Event-ID unterschiedlich gestaltet sein können, fällt diese Konfiguration sehr umfangreich aus. Für diesen Schritt wurden einzelne Pipeline-Konfigurationen aus dem Projekt „HELK“ [40] verwendet. Die eingesetzten Pipeline-Schritte sind in der Abbildung 4.3 ersichtlich.

Abbildung 4.3.: Überblick HELK Pipeline

Nachdem die Normalisierung und Filterung der Daten abgeschlossen wurde, werden die Log-Daten in den Elasticsearch-Cluster übertragen. Das Listing 4.33 zeigt die vollständige Ausgabekonfiguration. In diesem Fall wurden die Elasticsearch-Nodes erneut über Ansible dynamisch generiert.

```
{% set current_hosts = groups['tag_elastic'] | intersect(
   ansible_play_hosts_all|list) %}

  output {
    elasticsearch {
      hosts => [ {% for host in current_hosts | map('extract',
   hostvars, 'inventory_hostname') | list %}"https://{{ host
   }}:9200"{% if not loop.last %}, {% endif %}{% endfor %} ]
        index => "%{customtenant}_winevent_%{+YYYY.MM.dd}.log"
        ilm_enabled => "false"
        ssl_certificate_verification => "false"
        user => Benutzername
        password => Passwort
        template => "/usr/share/logstash/config/winevent.json"
        template_name => "winevent"
        template_overwrite => "true"
        manage_template => "true"
      }
```

Listing 4.33: Logstash - Pipeline Windows Event Output

4.2.3. Kibana

Für die Visualisierung der Daten und der später generierten Alarme wurde die Software Kibana verwendet. Diese Applikation kann die Informationen aus dem Elasticsearch-Cluster abrufen und für Benutzer/innen präsentieren. Für die Tactical-SIEM-Implementierung wurde Kibana als Docker-Container auf der vorgesehenen virtuellen Maschine mithilfe von Ansible installiert. Die Konfiguration der Kibana-Instanz wird mithilfe einer Konfigurationsdatei vorgenommen. Der Ausschnitt der Datei im Listing 4.34 zeigt die grundlegenden Einstellungen (Servername, TLS-Verschlüsselung und Zertifikate).

```
---
server.name: {{ inventory_hostname }}
server.host: 0.0.0.0
server.ssl.enabled: true
```

```
server.ssl.certificate: /usr/share/kibana/config/{{
   inventory_hostname }}.crt
server.ssl.key: /usr/share/kibana/config/{{ inventory_hostname
   }}.pem
server.ssl.keyPassphrase: {{ private_key_passphrase }}
server.ssl.certificateAuthorities: /usr/share/kibana/config/root
   .crt
elasticsearch.ssl.certificateAuthorities: /usr/share/kibana/
   config/root.crt
```

Listing 4.34: Kibana - Hostname und Zertifikate

Die Elasticsearch-Nodes, von welchen Kibana die notwendigen Daten auslesen kann, werden in der Konfigurationsdatei mithilfe von Ansible eingefügt. Dadurch ist es möglich, dass die Einstellung dynamisch an die vorhandene Infrastruktur angepasst wird. Im Konfigurationsauszug 4.35 ist die Ansible-Funktion und die Einstellungen für die Elasticsearch-Nodes ersichtlich.

```
{% set current_hosts = groups['tag_elastic'] | intersect(
   ansible_play_hosts_all|list) %}
  elasticsearch.hosts: {% for host in current_hosts | map('
   extract', hostvars, 'inventory_hostname') | list %}
    - "https://{{ host }}:9200" {# comment - gets all
   inventory_hostname with the tag tag_elastic #}
    {%- endfor %}
      elasticsearch.ssl.verificationMode: full
      elasticsearch.username: Elasticsearch-Benutzername
      elasticsearch.password: Passwort Elasticsearch-Benutzer
      elasticsearch.requestHeadersWhitelist: ["securitytenant","
   Authorization"]
```

Listing 4.35: Kibana - Elasticsearch Node Konfiguration

Der letzte Bestandteil, der eingesetzten Konfiguration, ist die Mandanten-Konfiguration für die Open-Distro-for-Elasticsearch-Version. Der Ausschnitt 4.36 zeigt die Aktivierung der Mandantenfunktion.

```
opendistro_security.multitenancy.enabled: true
opendistro_security.multitenancy.tenants.preferred: ["Tenant1"]
```

```
opendistro_security.multitenancy.enable_filter: true
opendistro_security.multitenancy.tenants.enable_global: true
opendistro_security.multitenancy.tenants.enable_private: true
```

Listing 4.36: Kibana - Mandanten Konfiguration

Für eine zusätzliche Absicherung der Kibana-Instanz, wurde „nginx“ als Reverse-Proxy implementiert. Auch diese Applikation wurden als Docker-Container bereitgestellt. Mithilfe des „nginx“ Reverse-Proxy wurden die TLS-Einstellungen vorgenommen und sicherheitsrelevante Header-Informationen eingefügt. Die eingesetzte „nginx“ Konfiguration befindet sich im Anhang A.4.

4.3. Anreicherung mithilfe von Open-Source-CTI

Als Quelle für die Open-Source-CTI wurde in der theoretischen Analyse die Software Malware Information Sharing Platform (MISP) ausgewählt. Die Installation der MISP-Instanz wurde ebenfalls innerhalb eines Docker-Containers und über Ansible durchgeführt. Da der MISP-Docker-Container auch für weitere Services verwendet werden kann, wird dieser nicht auf einer bestehenden virtuellen Maschine des Stacks betrieben, sondern eine eigenständige virtuellen Maschine für diesen Container verwendet.

4.3.1. Installation und Grundkonfiguration

Im ersten Schritt wird das notwendige Docker-Image erstellt und die Basiskonfiguration vorgenommen. Das Listing 4.37 zeigt die Erstellung des Docker-Images mithilfe von Ansible.

```
- name: build misp image
    docker_image:
    name: "{{ misp_image_name }}"
    build:
      path: "{{ misp_host_directory }}/repo/container"
      pull: yes
      args:
        MYSQL_MISP_PASSWORD: "{{ misp_mysql_password }}"
        POSTFIX_RELAY_HOST: "{{ misp_postfix_relay_host }}"
        MISP_FQDN: "{{ misp_fqdn }}"
        MISP_EMAIL: "{{ misp_email }}"
        MISP_GPG_PASSWORD: "{{ misp_gpg_password }}"
```

```
13      source: build
```

Listing 4.37: MISP Docker-Image Erstellung

Für eine Neu-Installation muss die Datenbank zuerst initialisiert werden. Dieser Schritt wird ebenfalls mithilfe des Docker-Container durchgeführt (Listing 4.38).

```
1  - name: init misp database
2      docker_container:
3      name: misp_db_init
4      image: "{{ misp_image_name }}"
5      auto_remove: yes
6      env:
7        TZ: Europe/Vienna
8      command: /init-db
9      volumes:
10       - misp_db:/var/lib/mysql
11   when: misp_db_volume.changed
12
13 - name: get misp_db_init status
14     docker_container_info:
15       name: misp_db_init
16     register: result
17     until: not result.exists
18     retries: 5
19     delay: 10
```

Listing 4.38: MISP Datenbank-Initialisierung

Nach Abschluss der Initialisierung wird der Docker-Container mit der Grundkonfiguration gestartet. Dieser Schritt wurde über das Ansible-Modul „docker_container“ durchgeführt und ist in folgendem Listing 4.39 ersichtlich.

```
1 - name: create misp container
2 docker_container:
3 name: misp
4 image: "{{ misp_image_name }}"
```

```
restart_policy: always
env:
TZ: Europe/Vienna
volumes:
- misp_db:/var/lib/mysql
- misp_config:/var/www/MISP/app/Config
- "/etc/ssl/{{ misp_fqdn }}/{{ misp_fqdn }}.pem:/etc/ssl/private
   /misp.crt"
- "/etc/ssl/{{ misp_fqdn }}/{{ misp_fqdn }}.key:/etc/ssl/private
   /misp.key"
published_ports:
- 80:80
- 443:443
tag: misp_{{ inventory_hostname }}
```

Listing 4.39: Start des Docker-Containers

Nach abgeschlossener Konfiguration und Start des Docker-Container, kann dieser über den definierten Hostnamen („misp_fqdn Variable“) per https aufgerufen werden.

4.3.2. Open-Source-Threat-Intelligence-Informationen

Damit die Installation Open-Source-Threat-Intelligence-Informationen erhält und wie geplant verteilen kann, müssen die erreichbaren Quell-Server definiert werden. Diese Quellen werden in der MISP-Software als „List-Feeds“ bezeichnet und können Informationen von angegeben Adressen in verschiedenen Formaten herunterladen. Zu diesen Formaten zählen:

- MISP-Feed - Dieses Format kann für den Austausch von Informationen zwischen mehreren MISP-Instanzen verwendet werden. In den übertragenen Informationen sind alle notwendigen Daten enthalten, damit die MISP-Events automatisch befüllt werden können.
- Freetext - Bei dem Freetext-Format handelt es sich um eine Möglichkeit, welche Textdateien (z.B. IP-Listen) importieren kann. Diese Daten werden in einem MISP-Event als Attribut hinzugefügt.
- Simple CSV - Dieses Format bietet ähnliche Funktionen wie das Freetext-Format. Der Unterschied hierbei ist, dass die CSV-Datei aus mehreren Spalten besteht und in der „List-Feed“ Konfiguration definiert werden kann, welche Spalten importiert werden sollen.

Nach der Installation der MISP-Software stehen zwei Standard-Quellen für Open-Source-Intelligence (OSINT) zur Verfügung.

- CIRCL OSINT Feed - Computer Incident Response Center Luxembourg
- The Botvrij.eu Data

Beide Quellen liefern umfangreiche Informationen über bekannte Angriffe bzw. Schadsoftware und stellen diese als Attribute in den MISP-Events zur Verfügung. Diese IOC können in der Applikation manuell ausgewertet oder über die API-Schnittstelle automatisiert verarbeitet werden. Zusätzlich zu den oben genannten Quellen wurde eine IP- (IPSum) und DNS-Sperrliste (Blackbook) als Freetext-Feed hinzugefügt. Mithilfe dieser Quellen wurden ungefähr 1400 MISP-Events importiert, welche insgesamt knapp 300.000 Attribute beinhalten. Für die automatische Erkennung im Tactical-SIEM werden aus diesen MISP-Events die Daten IP-Adresse, Hash-Werte, Hostnamen und Domänen ausgelesen. Für den Abruf dieser Informationen werden von der MISP-Software API-Funktionen angeboten, welche diese Attribute aus allen MISP-Events auswerten und im Text-Format retournieren. Aufrufbar sind diese Funktionen unter folgenden Pfaden:

IP-Adressen

- Quell-Adressen: /attributes/text/download/ip-src
- Ziel-Adressen: /attributes/text/download/ip-dst

Hash-Werte

- MD5 Hash: /attributes/text/download/md5
- SHA1 Hash: /attributes/text/download/sha1
- SHA256 Hash: /attributes/text/download/sha256

Hostnamen und Domänen

- Hostnamen: /attributes/text/download/hostname
- Domänen: /attributes/text/download/domain

4.3.3. Integration

Damit die gesammelten Informationen aus den MISP-Events auch für die Erkennung im Tactical-SIEM genutzt werden können, muss eine Möglichkeit verwendet werden welche, ohne größere Latenzen, einen

Vergleich zwischen den gesammelten Event-Log Informationen und den bekannten IOCs durchführt. Für diesen Anwendungsfall wurde eine Implementierung mit „memcache“ als Speicherort der IOC-Daten vorgesehen. Der Vorteil dieser Speicherung ist, dass die Daten nur im Arbeitsspeicher abgelegt werden und somit einen sehr schnellen Zugriff ermöglichen. Zusätzlich bietet „memcache“ eine Speicherung der Daten im Key/Value-Format, welche direkt für die Überprüfung verwendet werden kann. Der Abgleich zwischen den empfangenen Daten und den Informationen aus der MISP-Software wird innerhalb der jeweiligen Logstash-Pipeline durchgeführt. Für die Implementierung wurde ein Docker-Container für „memcache“ zusätzlich am virtuellen Logstash-Server installiert. Für die Erstellung des Docker-Image müssen keine speziellen Konfiguration berücksichtigt werden. Beim Start des Docker-Containers wurde der verfügbare Arbeitsspeicher von „memcache“ auf 1024 Megabyte fixiert (Zeile 6 bis 7 im Listing 4.40). Zusätzlich wurde der Dienst über das Netzwerk mit dem Port TCP 11211 erreichbar gemacht.

```
1  - name: create memcache container
2    docker_container:
3      name: "{{ inventory_hostname }}-memcache"
4      image: "{{ docker_memcache_image_name }}:{{
       docker_memcache_image_tag }}"
5      entrypoint:
6        - memcached
7        - -m 1024
8      networks:
9        - name: "{{ docker_network_name }}"
10     restart_policy: always
11     state: started
12     purge_networks: yes
13     published_ports:
14       - 11211:11211
```

Listing 4.40: memcache Docker-Container Start

Damit die Daten im „memcache“ abgerufen werden können, muss ein System diese aus den MISP-Events extrahieren und im „memcache“ speichern. Hierfür wurde ein Python-Skript erstellt, welches aus der MISP-Software, über die davor erwähnten API-Schnittstellen, die notwendigen Daten extrahiert. Das Listing 4.41 zeigt die dazu verwendete Funktion.

```
1  def getMispAttribute(dataType):
```

```
2   misp_api = misp_url + dataType
3   headers={'Authorization':misp_api_key,'Accept':'application/
     json','Content-type':'application/json'}
4   response = requests.get(misp_api,headers=headers,verify=False)
5   return response
```

Listing 4.41: Python Funktion für MISP Attributabfrage

In der Hauptfunktion des Python-Skripts wird je Datentyp die dazugehörige Funktion aufgerufen. Nachdem die Funktion die Informationen der MISP-Software retourniert, werden diese in den „memcache“ übertragen (Listing 4.42, Zeile 9). Die Hash-Werte werden davor in Großbuchstaben umgewandelt, damit diese direkt im Logstash verglichen werden können.

```
1 dataTypes={'domain', 'hostname', 'ip-dst', 'ip-src', 'md5', '
     sha256'}
2 client = Client((memcached_host, memcached_port))
3
4 for dataType in dataTypes:
5   response = getMispAttribute(dataType)
6   for line in response.text.splitlines():
7     if (dataType in ('md5', 'sha256')):
8       line = line.upper()
9     client.set(dataType + '-' + line, 'blacklist', 0)
```

Listing 4.42: Python memcache Integration

Die Daten im „memcache“ werden im Format „<Datentyp>-<Wert>:blacklist“ abgespeichert. Dieses Format wurde gewählt, damit bei späteren Abfragen der Typ des Wertes noch unterschieden werden kann (IP, MD5, SHA256). Als Wert im „memcache“ wird in dieser Version „blacklist“ eingetragen und bedeutet für die spätere Auswertung, dass sich dieser Wert auf einer Blacklist befindet. Es wäre auch möglich, hier weitere Informationen aus den MISP-Events zu extrahieren und abzuspeichern. Das vollständige Skript befindet sich im Anhang A.5.

Nach der Ausführung des Python-Skriptes befinden sich die aktuellen Daten der MISP-Events in dem angelegten „memcache“. Im nächsten Schritt müssen die empfangenen Event-Daten mit diesen Informationen verglichen werden. Damit dieser Vergleich möglich ist, wurde die Logstash-Pipeline erweitert. Bevor die Daten in den Elasticsearch-Cluster übertragen werden, wurde ein zusätzlicher Filter für die Überprüfung implementiert. Logstash bietet für diesen Anwendungsfall ein eigenes Filter-Plugin für einen Abgleich mit

einem „memcache“. Das Logstash-Plugin kann mithilfe der GET-Funktion einen Wert aus dem zurzeit verarbeiteten Feld im „memcache“ suchen. Wird der Wert nicht gefunden, wird in der Logstash-Pipeline der nächste Schritt durchgeführt. Bei einem positiven Abgleich wird der hinterlegte Wert aus dem „memcache“ retourniert. In diesem Fall wird „blacklist“ retourniert und im weiteren Schritt ein neues Feld „misp_info“ mit dem Wert befüllt.

Im Listing 4.43 wird die Überprüfung für die Datentypen Ziel-IP und Hash-Wert gezeigt. Es wird jeweils überprüft, ob das entsprechende Feld im erhaltenen Event-Log vorhanden ist und erst danach ein Abgleich mit dem „memcache“ durchgeführt. Da die MISP-Attribute auf Quell- und Ziel-IP-Adressen aufgeteilt sind, aber die Unterscheidung der beiden Typen nicht eindeutig ist (Quelle der Schadsoftware / Quelle im Netzwerkverkehr), wurden beide Typen für den Abgleich verwendet.

```
filter {
  if [dst_ip_addr] {
    memcached {
      hosts => [ "{{ inventory_hostname }}:11211" ]
      get => {
        "ip-dst-%{dst_ip_addr}" => "misp_info"
        "ip-src-%{dst_ip_addr}" => "misp_info"
      }
    }
  }
  if [hash_sha256] {
    memcached {
      hosts => [ "{{ inventory_hostname }}:11211" ]
      get => {
        "sha256-%{hash_sha256}" => "misp_info"
      }
    }
  }
}
```

Listing 4.43: Logstash Pipeline - memcache Filter-Plugin

Mithilfe des abgespeicherten Wertes im neuen Feld „misp_info“ können im Elasticsearch-Cluster weitere Auswertungen oder Alarmierungen implementiert werden.

4.4. Regelwerk Integration im Sigma-Format

Die Sigma-Regeln werden in einem Github-Projekt gewartet [41]. Dadurch kann das Regelwerk direkt von diesem Github-Projekt geklont und regelmäßig aktualisiert werden. Durch diese Vorgehensweise kann die GIT-Funktionalität, für eine Sicherstellung der Aktualität der Regeln, verwendet werden. Das Sigma-Regelwerk, welches von Github geklont bzw. aktualisiert wird, wird in den definierten YAML-Files heruntergeladen. Im Projekt Sigma wird auch das Programm „sigmac“ angeboten. Dieses kann dazu verwendet werden, um die Sigma Regeln im YAML-Format auf die gewünschte Syntax der Zielplattform konvertieren zu können. Für die Aktualisierung, Umwandlung und die Aktivierung der Regel wurde ein eigener Docker-Container mit einem Python-Skript gestartet. Dieser Container wird am Docker-Host für Kibana betrieben. Die Abbildung 4.4 zeigt die Aufgaben und den Ablauf des erstellten Python-Skripts.

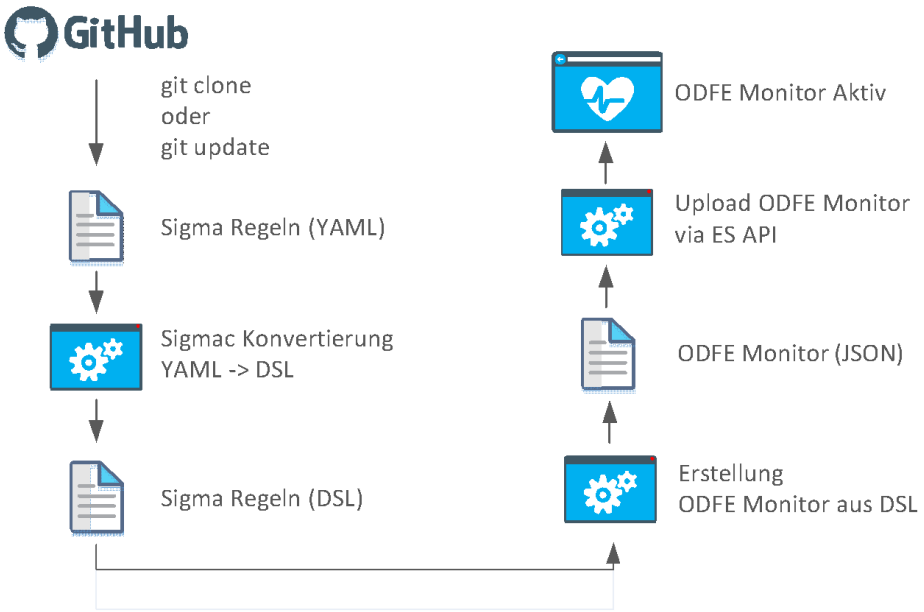

Abbildung 4.4.: Ablauf Python Sigma Konverter

Im ersten Schritt wird überprüft, ob das Sigma-Projekt bereits geklont wurde. Ist dies der Fall, wird ein Update des Projekts durchgeführt. Falls das Projekt noch nicht vorhanden ist, wird dieses von Github geklont. Durch diese Vorgehensweise befinden sich innerhalb des Containers die aktuellen Sigma-Regeln im YAML-Format.

Eine Erweiterung des Sigma-Regelwerks mit eigenen spezifischen Regeln ist über mehrere Wege möglich.

Eine Variante ist, dass das offizielle Projekt als „Fork“ in einem eigenen Projekt weitergepflegt wird. Dies würde voraussetzen, dass die Regeln mit der offiziellen Github-Plattform synchronisiert werden müssen. Falls diese Variante mit einer Online-Pflege nicht umsetzbar ist (z.B. Daten dürfen das lokale Netzwerk nicht verlassen), dann kann ein lokaler Gitlab-Server verwendet werden. Auf diesem Server werden die lokalen Regeln gepflegt und mithilfe des Skripts ebenfalls geklont.
Im weiteren Schritt wird mithilfe von „sigmac“ eine Umwandlung der Sigma-Regeln vorgenommen. Bei dieser Vorgehensweise muss darauf geachtet werden, dass eine entsprechende Konfiguration mit den Feldnamen hinterlegt ist. Aus dieser Konfigurationsdatei werden die Feldnamen entnommen, um die Regeln im YAML-Format korrekt auf die Regeln im notwendigen Elasticsearch-DSL-Format umzuwandeln. Dadurch können im Umwandlungsschritt die im Elasticsearch-Index verwendeten Feldnamen eingefügt werden. Das Listing 4.44 zeigt einen Auszug der Field-Mappings in der eingesetzten Konfigurationsdatei.

```
1 CallingProcessName: process_path
2 CallTrace: process_call_trace
3 ClientAddress: src_ip_addr
4 ClientIPAddress: src_ip_addr
5 ClientIP: src_original_value
6 CommandLine: process_command_line
7 Company: file_company
8 ComputerName: host_name
```

Listing 4.44: Sigmac - Feld Zuweisung

Die notwendigen Parameter für den Aufruf dieses Tools sind im Listing 4.45 ersichtlich.

```
1  sigmac -t <target> -c <configuration> -o <output-path> <source
     sigma rule>
```

Listing 4.45: Sigmac - Parameter für Programmaufruf

Bei der Umwandlung der Sigma-Regeln in das Elasticsearch-DSL-Format gibt es einzelne Fälle, welche von der DSL-Syntax nicht unterstützt werden. Dies beinhaltet z.B. Regeln welche eine „near“ Aggregation verwenden. Diese Aggregation kann dazu verwendet werden, um spezielle Abfragen zu generieren, welche nur das Ergebnis liefern, wenn zusätzlich zu einem gesuchten Wert eine weitere Abfrage entweder gefunden oder nicht gefunden wurde.
Regeln welche ein nicht unterstütztes Format für Elasticsearch-DSL beinhalten, erzeugen als Export eine leere Datei, welche nicht weiter verarbeitet werden kann. Aus diesem Grund muss das eingesetzte Python-Skript die erzeugten Dateien auf Inhalt überprüfen und in weitere Folge die leeren Dateien entfernen. Hierfür

wurde der gesamte Inhalt des Export-Verzeichnis durchlaufen und überprüft ob es sich um eine Datei mit Inhalt handelt. Das Listing 4.46 zeigt den verwenden Programmcode für die Überprüfung.

```
for dsl_rule in os.listdir(export_path):
    if ((not os.path.isfile(export_path + "/" + dsl_rule)) or\
    (os.path.getsize(export_path + "/" + dsl_rule) == 0)):
      # skip if file is a folder or file-size is 0
        continue
```

Listing 4.46: Sigmac - Python Überprüfung auf leere Dateien

Danach können die umgewandelten Regeln, welche zu diesem Zeitpunkt bereits im JSON-Format sind, weiterverarbeitet werden. Damit die Regeln für die Verwendung in einem ODES-Monitor vorbereitet werden können, ist es notwendig, dass das JSON-File mit den notwendigen Parametern der Monitorkonfiguration ergänzt wird. Für diesen Fall wurde eine Datei als JSON-Template für eine Monitorkonfiguration angelegt. Diese beinhaltet die notwendigen Abschnitte, damit der Monitor angelegt werden kann und konfiguriert einen Trigger, welcher bei einem Auftreten des gesuchten Events alarmiert. Die definierten Namen im Template werden im Python-Skript dynamisch anhand der konvertierten Regel gesetzt. Als Monitor-Intervall wurde fünf Minuten gewählt, da jede Ausführung eines Monitors eine Last auf dem Cluster verursacht und eine Erkennung eines Vorfalls innerhalb von fünf Minuten für diese Plattform ausreichend ist. Durch diesen Import ergibt sich eine Anzahl von ca. 600 aktiven Monitore. Im Listing 4.47 ist die erstelle Monitor-Template-Datei ersichtlich.

```
  {
    "type": "monitor",
    "name": "<monitor_name>",
    "enabled": true,
    "schedule": {
      "period": {
        "interval": 5,
        "unit": "MINUTES"
      }
    },
    "inputs": [{
      "search": {
```

```
        "indices": ["<index_name>"],
        "query": {

        }

      }
    }],
    "triggers": [{
      "name": "<monitor_trigger>",
      "severity": "1",
      "condition": {
        "script": {
          "source": "ctx.results[0].hits.total.value > 0",
          "lang": "painless"
        }
      },
      "actions" : [ ]
    }]
  }
```

Listing 4.47: Sigmac - Template Datei für neue Monitore

Im weiteren Verlauf des Python-Skripts wird das erstellte JSON-Template eingelesen und mit dem Inhalt der konvertierten Sigma-Regeln ergänzt. Hierfür wird der Ordner mit den exportierten Sigma-DSL-Regeln durchlaufen und der Name bzw. die DSL-Abfrage übernommen. Im nächsten Schritt werden die Daten einerseits als Datei im Filesystem exportiert und zusätzlich über die Elasticsearch-API hochgeladen. Diese Vorgehensweise ist im Listing 4.48 enthalten.

```
monitor_template = json.loads(mon_temp.read())
with open(export_path + "/" + dsl_rule, 'r') as f:
    json_rule = json.loads(f.read())
    monitor_template["inputs"][0]["search"]["query"] = json_rule
    monitor_template["name"] = dsl_rule
with open(export_path + "/odes_rules/" + dsl_rule, 'w+') as
   odes_export:
```

```
7     json.dump(monitor_template, odes_export)
8     r = requests.post(url, auth = HTTPBasicAuth(username,
    password),\
9     data=json.dumps(monitor_template), headers=headers)
```

Listing 4.48: Sigmac - Elasticsearch Anlage der Monitore

Nach einem erfolgreichen Durchlauf dieses Abschnitts befinden sich die angelegten Monitore im Elasticsearch-Index und sind über die Kibana-Oberfläche ersichtlich. Die Abbildung 4.5 zeigt einen Auszug der erstellen Monitore.

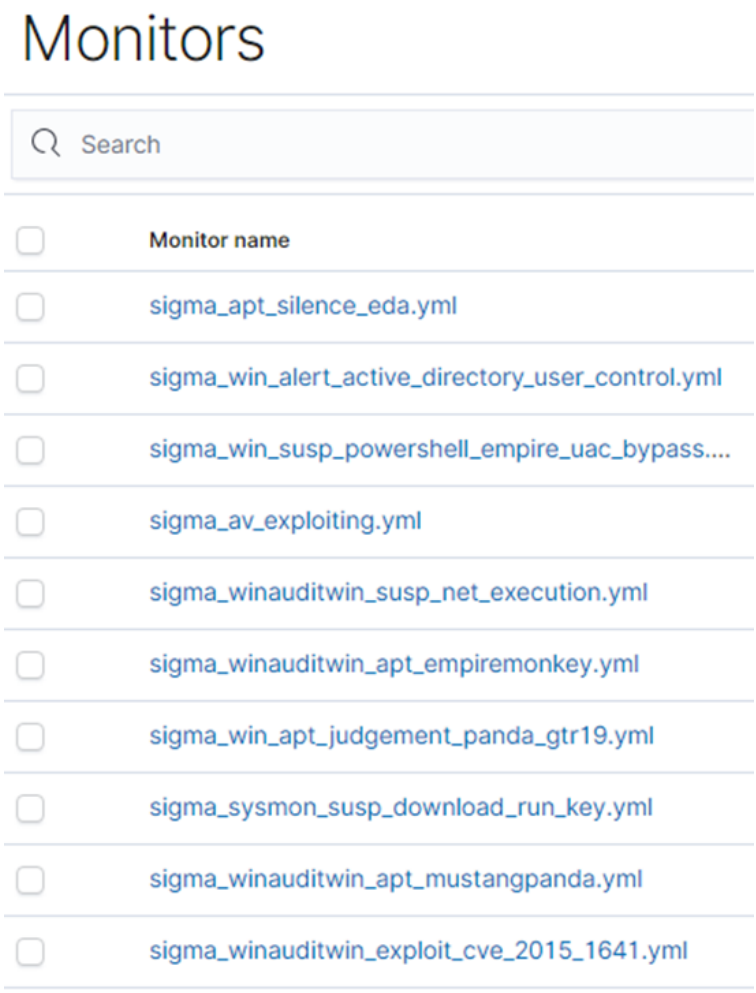

Abbildung 4.5.: Auszug Kibana Monitor

Notwendige Anpassung im Elasticsearch-Cluster

Die Standardwerte eines Elasticsearch-Cluster limitieren die Ausführung von dynamischen Skripten auf 75 Ausführungen pro fünf Minuten. Da die Abfragen in den Monitoren in diese Kategorie fallen, können nicht

alle Monitore ausgeführt werden. Die Elasticsearch-Instanz liefert in einem solchen Fall einen Fehler und protokolliert diesen in den Logs [42]. Im Listing 4.49 ist die generierte Fehlermeldung ersichtlich.

```
Exception: org.elasticsearch.common.breaker.
 CircuitBreakingException:
[script] Too many dynamic script compilations).
```

Listing 4.49: Elasticsearch - Fehlermeldung für zuviele Skript-Ausführungen

Der auszuführende Monitor erhält einen Error-State, welcher die Ergebnisse verfälschen kann. Für die Implementierung des Sigma Regelwerks wurde der Wert über die Elasticsearch-API auf 1000 dynamische-Skript-Kompilierungen pro fünf Minuten erhöht.

```
PUT /_cluster/settings
{
  "transient" : {
    "script.max_compilations_rate" : "1000/5m"
  }
}
```

Listing 4.50: Elasticsearch - Anpassung der maximalen Skript-Ausführungen

4.5. Logquellen-Konfiguration und Eventfilter

Für die Implementierung des Tactical-SIEMs im Zuge der Diplomarbeit werden die notwendigen Eventlogs von einem Windows Server 2019 Test-System generiert. Die Übertragung der Log-Daten übernimmt in diesem Fall die Software „Winlogbeat“. Hierbei handelt es sich um einen Agenten von elastic.co, welcher direkt auf die Windows-Eventlogs zugreifen und weiterleiten kann.
Nachdem der Agent mithilfe des angebotenen Installationspakets auf den Systemen aktiviert wurde, kann dieser mit einer Konfigurationsdatei angepasst werden. Die notwendige Datei definiert welche Log-Quellen der Agent auslesen soll. Diese Quellen können mit zusätzlichen Filter versehen werden. Das Listing 4.51 zeigt einen Auszug der Konfiguration, welche die notwendigen Windows-Events definiert. In dieser Definition sind ebenfalls Event-Filter enthalten. Mithilfe dieser werden nur Eventlogs ausgelesen, welche älter als 30 Minuten sind. Diese Zeitspanne wird für neue Windows-Events angewandt, damit bei einer Neuinstallation oder einer längeren Inaktivität der Software nicht der gesamte Windows-Eventlog-Inhalt übermittelt

wird. Eine bestehende aktive Installation erkennt welches Event zuletzt übertragen wurde und verarbeitet nur Log-Daten ab diesem Zeitpunkt. Zusätzlich werden in der Konfiguration bestimmte Log-Quellen auf Event-IDs gefiltert (z.B. Zeile 12 - Nur PowerShell-Script-Logging Informationen). Diese Filter dienen zur Minimierung der übertragenen Log-Daten und dadurch zu einer Reduzierung des benötigten Speichervolumens am zentralen Tactical-SIEM.

```
1  winlogbeat.event_logs:
2    - name: Application
3      ignore_older: 30m
4    - name: Security
5      ignore_older: 30m
6    - name: System
7      ignore_older: 30m
8    - name: Microsoft-windows-sysmon/operational
9      ignore_older: 30m
10   - name: Microsoft-windows-PowerShell/Operational
11     ignore_older: 30m
12     event_id: 4103, 4104
13   - name: Windows PowerShell
14     event_id: 400,600
15     ignore_older: 30m
16   - name: Microsoft-Windows-WMI-Activity/Operational
17     event_id: 5857,5858,5859,5860,5861
18     ignore_older: 30m
19   - name: Microsoft-Windows-Windows Defender/Operational
20     ignore_older: 30m
```

Listing 4.51: Winlogbeat - Eventlog-Konfiguration

Der zweite Bestandteil der Konfigurationsdatei ist die Einstellung der gewünschten Ausgabefunktion. In diesem Aufbau werden die ausgelesenen Eventlogs an die zentrale Logstash-Instanz gesendet. Hierfür wurde in der Konfigurationsdatei das Modul „output.logstash“ aktiviert und die gewünschte Ziel-Adresse definiert (4.52).

```
1  output.logstash:
2  enabled: true
```

```
3   hosts: ["<Hostname>:<port>"]
```

Listing 4.52: Winlogbeat - Ausgabe Konfiguration

4.5.1. Sysmon

Wie in der theoretischen Analyse (Kapitel 2.7.2 System-Monitor (sysmon)) ermittelt, muss für eine Erkennung von möglichen Angriffen oder einer Ausführung von Schadcode, eine zusätzliche Eventlog-Protokollierung mithilfe der Software „sysmon“ implementiert werden. Als Konfigurationsdatei für die Installation wurde die „SwiftOnSecurity sysmon-config“ [30] verwendet. Diese Datei beinhaltet bereits notwendige Ausnahmen für legitime Aufrufe, damit diese nicht protokolliert werden. Ohne dieser Konfigurationsdatei würde „sysmon“ alle Aktionen der Windows-Basisdienste bzw. von Standard-Programmen protokollieren und das Eventlog-Volumen unnötig vergrößern. Das Listing 4.53 zeigt, wie die Installation mit der ausgewählten Konfiguration durchgeführt werden kann.

```
1 sysmon.exe -accepteula -i sysmonconfig-export.xml
```

Listing 4.53: Sysmon Installation mit Konfiguration

Durch diesen Aufruf wird „sysmon“ installiert und automatisch als Windows-Dienst konfiguriert. Dadurch werden das Programm und die notwendigen Treiber automatisch bei jedem Systemstart gestartet. Die Protokollierung des Dienstes wird in das Windows-Eventlog integriert. Damit diese Daten an das zentrale System übermittelt werden, wurde in der „Winlogbeat“ Konfiguration der Pfad „Microsoft-windows-sysmon/operational“ angegeben.

4.5.2. PowerShell-Logging

Aufgrund der verbreiteten Angriffsvariante über die integrierten Funktionen der PowerShell, wurde das Script-Block-Logging am System aktiviert [43]. Details zu den unterschiedlichen Protokollierungsvarianten befinden sich im Kapitel 2.7.1 PowerShell-Logging. Das Listing 4.54 zeigt den notwendigen Registry-Schlüssel für die Aktivierung der PowerShell-Logging-Funktion.

```
1 HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\
     ScriptBlockLogging
2   EnableScriptBlockLogging = 1
```

Listing 4.54: PowerShell - Script-Block-Logging

Die aufgezeichneten Event-Logs werden in den Event-ID 4103 und 4104 protokolliert und durch den „Winlogbeat“ aufgrund der im Listing 4.55 ersichtlichen Konfiguration ausgelesen.

```
- name: Microsoft-windows-PowerShell/Operational
  ignore_older: 30m
  event_id: 4103, 4104
```

Listing 4.55: Winlogbeat - PowerShell Quelle

4.5.3. Import von vorhanden Event-Kollektionen

Damit die Funktionsweise der Erkennung und der Log-Verarbeitung getestet werden kann, ist es notwendig, die entsprechenden Log-Einträge zu erzeugen. Einige Logdaten, welche für die Alarmierung benötigt werden, können nur mithilfe eines durchgeführten Angriffs erzeugt werden. Aufgrund des Umfangs der erkannten Angriffe, kann ein Test, sehr Aufwändig ausfallen. Damit bestimmte Testfälle ausgeführt werden können, ist es zuvor notwendig, eine entsprechende Infrastruktur aufzubauen (Domain-Controller, Applikationsserver, Clients) [44].
Da nach jeder größeren Änderung am System dessen Funktionalität getestet werden soll, wurde eine Möglichkeit gesucht, welche es erlaubt, bereits aufgezeichnete Logdaten erneut in das System zu übertragen. Durch diese Variante kann der Aufwand für die Erzeugung der notwendigen Log-Daten minimiert werden. Für die Durchführung solcher Tests ist es notwendig, eine Möglichkeit für die erneute Übermittlung der aufgezeichneten Log-Daten bereitzustellen. Hierfür bietet sich die Software Winlogbeat an, da diese mithilfe einer Konfigurationsdatei für die gewünschten Log-Quellen und Zielsysteme konfiguriert werden kann. Es ist möglich, die Quelle für EVTX-Dateien (Windows-Log-Format) zu definieren. Die minimale Quellkonfiguration für diesen Anwendungsfall ist im Listing 4.56 ersichtlich.

```
winlogbeat.event_logs:
  - name: ${EVTX_FILE}
    no_more_events: stop
```

Listing 4.56: Winlogbeat Quellenkonfiguration

Damit die später importierten Log-Daten an das korrekte Zielsystem geleitet werden, muss in der Konfiguration die Ausgabe definiert werden. Für das Tactical-SIEM werden die Log-Daten des Winlogbeat-Agenten an die Logstash-Instanz gesendet. Die Konfiguration für die Logstash-Ausgabe ist im Listing 4.57 ersichtlich.

```
output.logstash:
  enabled: true
  hosts: ["<Hostname>:<Port>"]
```

Listing 4.57: Winlogbeat Ausgabekonfiguration

Damit die Alarmierung des Tactical-SIEM getestet werden kann, ist es notwendig, dass Logdaten mit den bekannten Angriffen importiert werden. Eine Möglichkeit dies durchzuführen ist, öffentlich verfügbare Beispiel Log-Kollektionen zu verwenden. Das Projekt „Windows EVTX Samples“ [45] beinhaltet 200 Log-Kollektionen, welche anhand der MITRE-ATT&CK-Matrix kategorisiert sind. Die angebotenen Dateien beinhalten die Log-Einträge, welche bei der Durchführung des Angriffs generiert wurden. Der notwendige Programmaufruf, damit diese Daten anhand der gewünschten Winlogbeat-Konfiguration verarbeitet werden, befindet sich im Listing 4.58.

```
  winlogbeat.exe -c <Pfad zu Konfigurationsdatei> -e -E
   EVTX_FILE="Pfad zu EVTX Datei"
```

Listing 4.58: Winlogbeat Ausführung für Event-Weiterleitung

Durch diese Vorgehensweise werden die Log-Daten durch den Winlogbeat an die Logstash-Installation gesendet und die definierte Pipeline angewandt.

4.5.4. Generierung der Log-Einträge

Eine weitere Möglichkeit für die Erzeugung der Log-Einträge ist eine Software, welche die gewünschten Angriffe automatisiert durchführt oder simuliert. Ein Vorteil dieser Variante ist, dass weitere Sicherheitsimplementierungen (z.B. Endpoint-Security) in den Test miteinbezogen werden können. Zusätzlich werden die Log-Einträge auf dem System neu generiert und es kann dadurch auch die Sysmon-Konfiguration getestet werden [46].

Ein Projekt, welches ohne zusätzliche Anforderungen, Testfälle für die n neun ATT&CK-Matrix-Taktiken anbietet, ist das Projekt „APTSimulator“ [47]. Bei diesem Simulator handelt es sich um ein Windows-Batch-Skript, welches bei der Ausführung von Angriffen, weitere Softwarekomponenten aufruft. Durch die Implementierung in einem Windows-Batch-Skript, kann die Software auf allen Windows-Systemen ausgeführt werden. Im Projekt sind alle notwendigen Programme enthalten, weswegen hier auf keine weiteren Abhängigkeiten geachtet werden muss. Der Simulator sollte immer auf einen dafür vorgesehen Test-System eingesetzt werden. Einige Testfälle simulieren das Verhalten eines Angriffs bzw. erstellen die entsprechenden Log-Einträge. Dennoch sind Testfälle inkludiert, welche am System schadhafte Dateien ablegen und

sicherheitsrelevante Einstellungen anpassen. Nachdem der Simulator über die Batch-Datei gestartet wurde, können die gewünschten Testfälle ausgewählt werden. Das Listing 4.59 zeigt die zur Verfügung stehenden Auswahlmöglichkeiten.

```
[0] RUN EVERY TEST
[1] Collection
[2] Command and Control
[3] Credential Access
[4] Defense Evasion
[5] Discovery
[6] Execution
[7] Lateral Movement
[8] Persistence
[9] Privilege Escalation
```

Listing 4.59: APTSimulator Testfälle

Durch Auswahl des gewünschten Test-Falles werden die weiteren Schritte automatisch durchgeführt. Diese Schritte beinhaltet je nach Test-Fall einen automatischen Download von Schadsoftware und deren Ausführung.

5. Schlussfolgerung und Ausblick

In der vorliegenden Diplomarbeit wurde eine Analyse-Plattform auf Basis von Open-Source-Software implementiert. Die Implementierung bietet die Möglichkeit, eine Unterteilung auf Mandanten umzusetzen und dadurch eine Datentrennung für mehrere Umgebungen bereitzustellen. Durch die Docker-Container-Installationsvariante, welche mithilfe von Ansible verwaltet wird, kann das System in kürzester Zeit mit alternativen Versionen und Anforderungen bereitgestellt werden. Durch diese Funktionalität ist ebenfalls eine Ressourcenerweiterung der Plattform automatisiert umsetzbar.

Eine Anreicherung und Überprüfung der empfangenen Daten anhand Open-Source-Threat-Intelligence wurde mithilfe von Logstash und einer memcache-Instanz umgesetzt. Als Quellsystem für diese Informationen wurde die Software Malware-Information-Sharing-Platform (MISP) implementiert. Durch diese Software ist es möglich, Quellen für Indicators-of-Compromise (IOC) abzurufen und diese zu importieren. Da durch diese Variante alle Informationen in der MISP-Software gesammelt werden, muss bei der Verwendung dieser IOCs in weiteren Produkten nur die MISP-Schnittstelle betrachtet werden. Da bei einem positiven Ergebnis der Logstash-Überprüfung ein neues Feld zu den übermittelten Log-Daten hinzugefügt wird, kann in weiterer Folge eine generische Alarmierung implementiert werden. Durch diese generische Regel sind bei Änderungen der IOC-Informationen keine Anpassungen des Regelwerks notwendig und erfordern somit keinen manuellen Eingriff für die korrekte Erkennung.

Für die Alarmierung von komplexen Angriffsmustern wurden Open-Source-SIEM-Regeln im Sigma-Format integriert. Durch dieses Format können SIEM-Regeln mit einer generischen Syntax beschrieben werden. Dies ermöglicht es, dass Erkennungsregeln von möglichen Angriffen nicht auf bestimmte Produkte bezogen sind und zwischen Parteien ausgetauscht werden können. Damit die Regeln in einem SIEM verwendet werden können, müssen diese auf das gewünschte Format umgewandelt werden. Im Zuge des Aufbaus konnte festgestellt werden, dass spezielle Konditionen in den Regeln für bestimmte Zielsysteme nicht umgesetzt werden können bzw. nicht zu dem gewünschten Ergebnis führen. Dies hat zufolge, dass für eine automatisierte Konfiguration und Aktualisierung der Regeln ein manueller Kontrollschritt durchgeführt werden muss.

Damit eine Analyse der empfangenen Eventdaten durchgeführt werden kann, muss in einer Windows-Umgebung eine Sysmon-Konfiguration implementiert werden. Die Sysmon-Funktionalität bietet ein wei-

tes Spektrum an Protokollierungsmöglichkeiten, welche für die Integration benötigt werden. Im Zuge der Implementierungen hat sich gezeigt, dass ein Großteil der Informationen mithilfe von Sysmon gesammelt werden kann. Die Konfiguration von weiteren Audit-Funktionen kann somit auf wenige Einstellungen eingeschränkt werden.

Eine automatisierte Kategorisierung anhand der MITRE-ATT&CK-Matrix konnte nicht umgesetzt werden. Dennoch werden die Informationen der Matrix in den einzelnen Komponenten verwendet. In den Sigma-Regeln werden die dazugehörigen Techniken verknüpft, welche wiederum bei den Alarmen für die zuständigen Personen zur Verfügung stehen. Die idente Vorgehensweise ist bei der Sysmon-Protokollierung vorhanden. Die verwendete Sysmon-Konfiguration beinhaltet im Regelwerk eine Zuteilung zu den entsprechenden Techniken oder Kampagnen.

Damit die Skalierbarkeit und Modularität der Plattform auch für weitere Integrationen von Kundenumgebungen und Funktionserweiterungen gewährleistet ist, wurde bei der Implementierung eine Kombination von Ansible und Docker-Container verwendet. Die Verwendung von Ansible hat gezeigt, dass die Implementierung einen höheren Aufwand verursacht hat, aber der Betrieb und die Wartungen der Plattform dadurch ohne komplexe Dokumentationen möglich sind. Notwendige Ressourcenerweiterungen können mithilfe von Ansible-Playbooks im laufenden Betrieb durchgeführt werden.

Der Praxiseinsatz des Systems hat wie erwartet gezeigt, dass manuelle Anpassungen am Regelwerk je nach Kundenumgebung notwendig sind. Da jede Umgebung unterschiedlich aufgebaut und implementiert ist, sind Prozessabläufe vorhanden, welche im Regelwerk speziell definiert werden müssen. Eine unerwartete positive Erkenntnis in diesem Bereich ist, dass der Aufwand für die False-Positive-Bereinigung ab einer bestimmten Anzahl von Systemen nicht mehr steigt. In der Praxis hat sich bei der Implementierung von mehreren Kunden gezeigt, dass die Steigerung des Zeitaufwands ab 50 Servern vernachlässigbar ist. Der produktive Einsatz des Systems wurde mit einer Kundenumgebung im Parallelbetrieb zu einem bestehenden SIEM gestartet. In diesem Zeitraum wurde festgestellt, dass das Tactical-SIEM komplexere Angriffe durch das Sigma-Regelwerk erkennt. Dadurch ist die Bewertung der Alarme ebenfalls komplexer und erfordert Personal mit entsprechendem Wissen. Des Weiteren wurde in diesem Setup auch festgestellt, dass das bestehende SIEM unterschiedliche Alarme für System-Anmeldungen generiert hat, welche im Sigma-Regelwerk nicht abgebildet sind (z.B. Admin-Login oder fehlgeschlagene Anmeldungen). Diese Alarmierungsregeln müssen bei Bedarf manuell implementiert werden.

5.1. Weiterführende Arbeiten

Durch den produktiven Einsatz des Tactical-SIEMs wurde festgestellt, dass die Alarmierung für die Bearbeitung nicht ausreichend ist. Bei der Analyse von Alarmen benötigt ein/e Analyst/in die Möglichkeit, die Vorfälle zu dokumentieren. Aus diesem Grund sollten die generierten Alarme inklusive der betroffenen Eventlogs an ein weiteres System mit Dokumentationsmöglichkeit weitergegeben werden. Dadurch können bearbeitete Fälle auch archiviert und für spätere Folgetätigkeiten weiterverwendet werden. Damit eine Bewertung der Alarme erleichtert wird, sollte ein Hardware- und Softwareinventar der überwachten Systeme integriert werden. Zusätzlich zu diesen Informationen sind Ergebnisse von Schwachstellenüberprüfungen eine hilfreiche Ergänzung für die weitere Analyse. Durch solche Integrationen sollte eine automatische Verknüpfung zwischen einem Alarm, dem betroffenen System und möglichen Schwachstellen hergestellt werden.

Durch die Weiterentwicklung von Angriffswegen kann das zentrale SIEM neue Technologien mit dem statischen Regelwerk nicht erkennen. Erst nachdem eine entsprechende Sigma-Regel für diese Technologie entworfen wurde, wird diese automatisch integriert. Aus diesem Grund sollte eine automatische Anomaly-Erkennung mithilfe von Machine-Learning in Betracht gezogen werden. Die eingesetzte Open-Distro-for-Elasticsearch-Variante beinhaltet den k-NN-Algorithmus, welcher für eine Erkennung evaluiert werden sollte.

In der derzeitigen Implementierung besteht die Gefahr, dass ein Ausfall der Logstash-Instanz einen direkten Verlust von Event-Logs bedeutet. Je nach implementiertem Compliance-SIEM kann dieses die Daten bei Nichterreichbarkeit der Logstash-Instanz zwischenspeichern und erneut senden oder die Daten werden verworfen. Da das Konzept des Tactical-SIEM vorsieht, dass das zentrale SIEM möglichst unabhängig von den dezentralen Services arbeiten kann, sollte hier eine Möglichkeit für einen redundanten Eintrittspunkt der empfangenen Event-Daten geschaffen werden.

Die Verwendung von Angriffssimulatoren oder aufgezeichneten Event-Daten ermöglicht es zwar, dass die Funktionalität der Plattform überprüft werden kann, erfordert aber einen manuellen Arbeitsschritt für die Durchführung. Für eine regelmäßige Funktionskontrolle und Überwachung des Systems ist eine automatische Durchführung der Simulationen notwendig. Damit ein mögliches Problem erkannt werden kann, muss bei der automatischen Durchführung bereits definiert sein, welche Ergebnisse erwartet werden. Eine derartige Implementierung könnte regelmäßig ausgeführt werden und würde ein mögliches Problem zeitnah erkennen. Zusätzlich kann diese Funktionalität mit dem Ansible-Installations- und Aktualisierungsablauf verknüpft werden.

A. Anhang

A.1. Elasticsearch - Konfigurationsdatei

```
---
node.name: {{ inventory_hostname }}
node.master: true
node.data: true
node.ingest: true
cluster.name: {{ elastic_cluster_name }}
network.host: 0.0.0.0
network.publish_host: {{ inventory_hostname }}
bootstrap.memory_lock: true
{% set current_hosts = groups['tag_elastic'] | intersect(
   ansible_play_hosts_all|list) %}
discovery.seed_hosts: {% for host in current_hosts | map('
   extract', hostvars, 'inventory_hostname') | list %}
- {{ host }} {# comment - gets all inventory_hostname with the
   tag tag_elastic #}
{%- endfor %}

cluster.initial_master_nodes: {% for host in current_hosts | map
   ('extract', hostvars, 'inventory_hostname') | list %}
- {{ host }} {# comment - gets all inventory_hostname with the
   tag tag_elastic #}
{%- endfor %}

opendistro_security.ssl.transport.pemkey_password: {{
   private_key_passphrase }}
```

```
20 opendistro_security.ssl.transport.pemkey_filepath: {{
      inventory_hostname }}.pem
21 opendistro_security.ssl.transport.pemcert_filepath: {{
      inventory_hostname }}.crt
22 opendistro_security.ssl.transport.pemtrustedcas_filepath: root.
      crt
23 opendistro_security.ssl.http.enabled: true
24 opendistro_security.ssl.http.pemkey_password: {{
      private_key_passphrase }}
25 opendistro_security.ssl.http.pemkey_filepath: {{
      inventory_hostname }}.pem
26 opendistro_security.ssl.http.pemcert_filepath: {{
      inventory_hostname }}.crt
27 opendistro_security.ssl.http.pemtrustedcas_filepath: root.crt
28 opendistro_security.allow_unsafe_democertificates: false
29
30 opendistro_security.authcz.admin_dn:
31 - "CN=elastic-admin,O=anlx.cloud,C=AT"
32 opendistro_security.nodes_dn:
33 - "CN=*-Docker,O=anlx.cloud,C=AT"
34 opendistro_security.audit.type: internal_elasticsearch
35 opendistro_security.restapi.roles_enabled: ["all_access", "
      security_rest_api_access"]
```

A.2. Sigma Regel - Windows Event IDs

EventID	Beschreibung
614	IPSec policy agent disabled
624	User Account Created
625	User Account Type Changed
634	Security Enabled Global Group Deleted
646	Computer Account Changed
656	Security Disabled Global Group Member Removed
657	Security Disabled Global Group Deleted
658	Security Enabled Universal Group Created
660	Security Enabled Universal Group Member Added
661	Security Enabled Universal Group Member Removed
663	Security Disabled Universal Group Created
675	Pre-authentication failed
694	LDAP Query Group Created
852	A change has been made to the Windows Firewall port exception list
4611	A trusted logon process has been registered with the Local Security Authority
4616	The system time was changed.
4624	An account was successfully logged on
4625	An account failed to log on
4656	A handle to an object was requested
4657	A registry value was modified
4661	A handle to an object was requested
4662	An operation was performed on an object
4673	A privileged service was called
4674	An operation was attempted on a privileged object
4692	Backup of data protection master key was attempted
4698	A scheduled task was created
4704	A user right was assigned
4706	A new trust was created to a domain
4719	System audit policy was changed
4720	A user account was created

4732	A member was added to a security-enabled local group
4738	A user account was changed
4769	A Kerberos service ticket was requested
4794	An attempt was made to set the Directory Services Restore Mode administrator password
4825	A user was denied the access to Remote Desktop
5136	A directory service object was modified
5140	A network share object was accessed
5145	A network share object was checked to see whether client can be granted desired access
5156	The Windows Filtering Platform has allowed a connection
6416	A new external device was recognized by the system.

A.3. Sysmon Event-IDs

Folgende Dokumentation der Event-IDs wurde aus der offiziellen Microsoft-Dokumentation übernommen [29].

Event ID 1: Process creation

The process creation event provides extended information about a newly created process. The full command line provides context on the process execution. The ProcessGUID field is a unique value for this process across a domain to make event correlation easier. The hash is a full hash of the file with the algorithms in the HashType field.

Event ID 2: A process changed a file creation time

The change file creation time event is registered when a file creation time is explicitly modified by a process. This event helps tracking the real creation time of a file. Attackers may change the file creation time of a backdoor to make it look like it was installed with the operating system. Note that many processes legitimately change the creation time of a file; it does not necessarily indicate malicious activity.

Event ID 3: Network connection

The network connection event logs TCP/UDP connections on the machine. It is disabled by default. Each connection is linked to a process through the ProcessId and ProcessGUID fields. The event also contains the source and destination host names IP addresses, port numbers and IPv6 status.

Event ID 4: Sysmon service state changed

The service state change event reports the state of the Sysmon service (started or stopped).

Event ID 5: Process terminated

The process terminate event reports when a process terminates. It provides the UtcTime, ProcessGuid and ProcessId of the process.

Event ID 6: Driver loaded

The driver loaded events provides information about a driver being loaded on the system. The configured hashes are provided as well as signature information. The signature is created asynchronously for performance reasons and indicates if the file was removed after loading.

Event ID 7: Image loaded

The image loaded event logs when a module is loaded in a specific process. This event is disabled by default and needs to be configured with the –l option. It indicates the process in which the module is loaded, hashes and signature information. The signature is created asynchronously for performance reasons and indicates if the file was removed after loading. This event should be configured carefully, as monitoring all image load events will generate a large number of events.

Event ID 8: CreateRemoteThread

The CreateRemoteThread event detects when a process creates a thread in another process. This technique is used by malware to inject code and hide in other processes. The event indicates the source and target process. It gives information on the code that will be run in the new thread: StartAddress, StartModule and StartFunction. Note that StartModule and StartFunction fields are inferred, they might be empty if the starting address is outside loaded modules or known exported functions.

Event ID 9: RawAccessRead

The RawAccessRead event detects when a process conducts reading operations from the drive using the . denotation. This technique is often used by malware for data exfiltration of files that are locked for reading, as well as to avoid file access auditing tools. The event indicates the source process and target device.

Event ID 10: ProcessAccess

The process accessed event reports when a process opens another process, an operation that's often followed by information queries or reading and writing the address space of the target process. This enables detection of hacking tools that read the memory contents of processes like Local Security Authority (Lsass.exe) in order to steal credentials for use in Pass-the-Hash attacks. Enabling it can generate significant amounts of logging if there are diagnostic utilities active that repeatedly open processes to query their state, so it generally should only be done so with filters that remove expected accesses.

Event ID 11: FileCreate

File create operations are logged when a file is created or overwritten. This event is useful for monitoring autostart locations, like the Startup folder, as well as temporary and download directories, which are common places malware drops during initial infection.

Event ID 12: RegistryEvent (Object create and delete)

Registry key and value create and delete operations map to this event type, which can be useful for monitoring for changes to Registry autostart locations, or specific malware registry modifications.

Sysmon uses abbreviated versions of Registry root key names, with the following mappings:

EVENT ID 12: REGISTRYEVENT (OBJECT CREATE AND DELETE)

Key name	**Abbreviation**
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_LOCAL_MACHINE\System\ControlSet00x	HKLM\System\CurrentControlSet
HKEY_LOCAL_MACHINE\Classes	HKCR

Event ID 13: RegistryEvent (Value Set)

This Registry event type identifies Registry value modifications. The event records the value written for Registry values of type DWORD and QWORD.

Event ID 14: RegistryEvent (Key and Value Rename)

Registry key and value rename operations map to this event type, recording the new name of the key or value that was renamed.

Event ID 15: FileCreateStreamHash

This event logs when a named file stream is created, and it generates events that log the hash of the contents of the file to which the stream is assigned (the unnamed stream), as well as the contents of the named stream. There are malware variants that drop their executables or configuration settings via browser downloads, and this event is aimed at capturing that based on the browser attaching a Zone.Identifier “mark of the web” stream.

Event ID 16: ServiceConfigurationChange

This event logs changes in the Sysmon configuration - for example when the filtering rules are updated.

Event ID 17: PipeEvent (Pipe Created)

This event generates when a named pipe is created. Malware often uses named pipes for interprocess communication.

Event ID 18: PipeEvent (Pipe Connected)

This event logs when a named pipe connection is made between a client and a server.

Event ID 19: WmiEvent (WmiEventFilter activity detected)

When a WMI event filter is registered, which is a method used by malware to execute, this event logs the WMI namespace, filter name and filter expression.

Event ID 20: WmiEvent (WmiEventConsumer activity detected)

This event logs the registration of WMI consumers, recording the consumer name, log, and destination.

Event ID 21: WmiEvent (WmiEventConsumerToFilter activity detected)

When a consumer binds to a filter, this event logs the consumer name and filter path.

Event ID 22: DNSEvent (DNS query)

This event generates when a process executes a DNS query, whether the result is successful or fails, cached or not. The telemetry for this event was added for Windows 8.1 so it is not available on Windows 7 and earlier.

Event ID 23: FileDelete (A file delete was detected)

A file was deleted

Event ID 255: Error

This event is generated when an error occurred within Sysmon. They can happen if the system is under heavy load and certain tasked could not be performed or a bug exists in the Sysmon service.

A.4. Konfigurationsdatei - nginx

```
server {
  listen 8080;
  server_name 127.0.0.1;
  return 301 https://$host$request_uri;
}
server {
  listen 8443 ssl;
  ssl_certificate /etc/nginx/nginxcert.pem;
  ssl_certificate_key /etc/nginx/nginxkey.pem;
  server_tokens off;
  add_header X-Frame-Options DENY;
  add_header X-Content-Type-Options nosniff;
  add_header X-XSS-Protection "1; mode=block";
  add_header X-Frame-Options "SAMEORIGIN";
  client_body_buffer_size  100K;
  client_header_buffer_size 1k;
  client_max_body_size 7500k;
  ssl_ciphers "'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
   GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
   POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256";
  ssl_prefer_server_ciphers on;
  add_header Strict-Transport-Security "max-age=31536000;
   includeSubdomains;";
  proxy_connect_timeout 900;
  proxy_send_timeout 600;
  proxy_read_timeout 600;
  client_body_timeout   10;
  client_header_timeout 10;
  keepalive_timeout     10 10;
  send_timeout 10;
  gzip             on;
  gzip_comp_level  1;
```

```
30    gzip_min_length  1000;
31    gzip_proxied     expired no-cache no-store private auth;
32    gzip_types text/plain text/css application/json application/x-
       javascript text/xml application/xml application/xml+rss text/
       javascript;
33    large_client_header_buffers 4 64k;
34    location / {
35      proxy_pass https://{{ inventory_hostname }}:5601;
36      proxy_http_version 1.1;
37      proxy_set_header Upgrade $http_upgrade;
38      proxy_set_header Connection 'upgrade';
39      proxy_set_header Host {{ inventory_hostname }};
40      proxy_cache_bypass $http_upgrade;
41    }
42  }
```

A.5. Anreicherung - Python Skript für MISP Informationen

Folgendes Skript wurde über Ansible auf den Zielhost übertragen. Damit die notwendigen Serveradressen für die jeweilige Umgebung korrekt eingetragen sind, wurde ein Ansible-Template mit Ansible-Variablen verwendet.

```
import requests
from requests.packages.urllib3.exceptions import
   InsecureRequestWarning
from pymemcache.client.base import Client
import json
requests.packages.urllib3.disable_warnings(
   InsecureRequestWarning)

memcached_host = "{% for host in (groups['tag_memcache'] | map('
   extract', hostvars, 'inventory_hostname') | list) %}{{ host
   }}{%- endfor %}"
memcached_port = 11211
misp_url = "https://{{ misp_fqdn }}/attributes/text/download/"
misp_api_key = "{{ misp_api_key }}"

client = Client((memcached_host, memcached_port))
dataTypes={'domain', 'hostname', 'ip-dst', 'ip-src', 'md5', '
   sha256'}

def getMispAttribute(dataType):
  misp_api = misp_url + dataType
  headers={'Authorization':misp_api_key,'Accept':'application/
   json','Content-type':'application/json'}
  response = requests.get(misp_api,headers=headers,verify=False)
  return response

def main():
  for dataType in dataTypes:
    response = getMispAttribute(dataType)
```

```
24      for line in response.text.splitlines():
25        if (dataType in ('md5', 'sha256')):
26          line = line.upper()
27        client.set(dataType + '-' + line, 'blacklist', 0)
28        print(line)
29
30  if __name__ == '__main__':
31    main()
```

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[1] (2020) M-Trends 2020. Fireeye Mandiant Services. [Online]. Available: https://www.fireeye.com/content/dam/collateral/en/mtrends-2020.pdf [zugegriffen am: 15.01.2021]

[2] (2020) Cost of a Data Breach Report 2020. Ponemon Institute LLC. [Online]. Available: https://www.capita.com/sites/g/files/nginej146/files/2020-08/Ponemon-Global-Cost-of-Data-Breach-Study-2020.pdf [zugegriffen am: 15.01.2021]

[3] Mavroeidis, Vasileios and Bromander, Siri, "Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence," 09 2017, pp. 91–98.

[4] (2019) The cost of cybercrime. Ponemon Institute LLC. [Online]. Available: https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf [zugegriffen am: 12.11.2020]

[5] M. Fuchs and J. Lemon, "SANS 2020 Threat Hunting Survey Results," 12 2020. [Online]. Available: https://www.sans.org/reading-room/whitepapers/analyst/2020-threat-hunting-survey-results-40020 [zugegriffen am: 16.02.2021]

[6] (2020) ENISA Threat Landscape 2020 - Cyber threat intelligence overview. European Union und Agency for Network and Information Security. [Online]. Available: https://www.enisa.europa.eu/publications/cyberthreat-intelligence-overview/at_download/fullReport [zugegriffen am: 15.02.2021]

[7] F. Janos and P. D. Nguyen, "Security Concerns Towards Security Operations Centers," 05 2018, pp. 000 273–000 278.

[8] O. Procopiuc, P. Agarwal, L. Arge, and J. Vitter, "Bkd-Tree: A Dynamic Scalable kd-Tree," 07 2003, pp. 46–65.

[9] S. Sanjappa and M. Ahmed, "Analysis of logs by using Logstash," in *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*. Springer, 2017, pp. 579–585.

[10] E. Glass. Installieren von Elasticsearch, Logstash und Kibana (Elastic Stack) unter Ubuntu 20.04. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-20-04-de [zugegriffen am: 14.02.2021]

[11] "Logstash Filter Plugins." [Online]. Available: https://www.elastic.co/guide/en/logstash/current/filter-plugins.html [zugegriffen am: 10.01.2021]

[12] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[13] F. Soppelsa and C. Kaewkasi, *Native Docker Clustering with Swarm*. Packt Publishing Ltd, 2016.

[14] M. F. Haque and R. Krishnan, "Toward Automated Cyber Defense with Secure Sharing of Structured Cyber Threat Intelligence," *Information Systems Frontiers*, pp. 1–14, 2021.

[15] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 2016, pp. 49–56.

[16] "Projekt ipsum." [Online]. Available: https://github.com/stamparm/ipsum [zugegriffen am: 10.01.2021]

[17] "Projekt Blackbook." [Online]. Available: https://github.com/stamparm/blackbook [zugegriffen am: 13.01.2021]

[18] "Kaspersky Threat Data Feeds." [Online]. Available: https://tip.kaspersky.com/help/Doc_data/ThreatDataFeeds.htm [zugegriffen am: 10.01.2021]

[19] "Cybercure Blocked Hash API." [Online]. Available: https://docs.cybercure.ai/docs/blocked-hashes-api [zugegriffen am: 10.01.2021]

[20] "Virustotal File API." [Online]. Available: https://developers.virustotal.com/reference#file-feed [zugegriffen am: 10.01.2021]

[21] T. Patzke. (2017) Threat Hunting with Application Logs and Sigma. [Online]. Available: https://owasp.org/www-pdf-archive/GOD17-Sigma.pdf [zugegriffen am: 13.12.2020]

[22] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," *Technical report*, 2018.

[23] B. E. Strom, J. A. Battaglia, M. S. Kemmerer, W. Kupersanin, D. P. Miller, C. Wampler, S. M. Whitley, and R. D. Wolf, "Finding cyber threats with ATT&CK-based analytics," *The MITRE Corporation, Bedford, MA, Technical Report No. MTR170202*, 2017.

[24] V. S. M. Legoy, "Retrieving ATT&CK tactics and techniques in cyber threat reports," Master's thesis, University of Twente, 2019.

[25] S. M. Pontiroli and F. R. Martinez, "The Tao of .NET and PowerShell Malware Analysis," in *Virus Bulletin Conference*, 2015.

[26] R. Kazanciyan and M. Hastings, "Investigating PowerShell Attacks," *Black Hat*, p. 25, 2014.

[27] Fireeye Powershell Logging. [Online]. Available: https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html [zugegriffen am: 11.06.2021]

[28] V. Mavroeidis and A. Jøsang, "Data-driven threat hunting using sysmon," in *Proceedings of the 2nd International Conference on Cryptography, Security and Privacy*, 2018, pp. 82–88.

[29] "Sysmon v13.00 Documentation." [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon [zugegriffen am: 10.01.2021]

[30] "sysmon-config | A Sysmon configuration file for everybody to fork." [Online]. Available: https://github.com/SwiftOnSecurity/sysmon-config [zugegriffen am: 10.01.2021]

[31] "sysmon-modular | A Sysmon configuration repository for everybody to customise." [Online]. Available: https://github.com/olafhartong/sysmon-modular [zugegriffen am: 14.01.2021]

[32] J. Keating, *Mastering Ansible*. Packt Publishing Ltd, 2015.

[33] "Elasticsearch - Docker production mode." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/5.1/docker.html#docker-cli-run-prod-mode [zugegriffen am: 14.01.2021]

[34] "Documentation for /proc/sys/vm/*." [Online]. Available: https://www.kernel.org/doc/Documentation/sysctl/vm.txt [zugegriffen am: 14.01.2021]

[35] "Configuring Elasticsearch: Elasticsearch Reference [7.10]." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/settings.html [zugegriffen am: 10.01.2021]

[36] "Configuring Elasticsearch Logging." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/master/logging.html [zugegriffen am: 10.01.2021]

[37] "Elasticsearch - Tune index speed." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/tune-for-indexing-speed.html [zugegriffen am: 13.01.2021]

[38] "Sizing Amazon ES Domains." [Online]. Available: https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/sizing-domains.html#aes-bp-sharding [zugegriffen am: 10.01.2021]

[39] "Elasticsearch - Dynamic field mapping." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/dynamic-field-mapping.html [zugegriffen am: 14.01.2021]

[40] "The Hunting ELK." [Online]. Available: https://github.com/Cyb3rWard0g/HELK [zugegriffen am: 10.01.2021]

[41] "Sigma." [Online]. Available: https://github.com/Neo23x0/sigma [zugegriffen am: 14.01.2021]

[42] "Elasticsearch - Script comilation circuit breaker." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/7.3/circuit-breaker.html#script-compilation-circuit-breaker [zugegriffen am: 14.01.2021]

[43] D. Bohannon and L. Holmes. Revoke-Obfuscation: PowerShell Obfuscation Detection Using Science. [Online]. Available: https://www.fireeye.com/content/dam/fireeye-www/blog/pdfs/revoke-obfuscation-report.pdf [zugegriffen am: 11.03.2021]

[44] A. Bhardwaj and S. Goundar, "A framework for effective threat hunting," *Network Security*, vol. 2019, no. 6, pp. 15–19, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1353485819300741 [zugegriffen am: 09.02.2021]

[45] "Windows EVTX Samples." [Online]. Available: https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES [zugegriffen am: 14.01.2021]

[46] S. Randhawa, B. Turnbull, J. Yuen, and J. Dean, "Mission-Centric Automated Cyber Red Teaming," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ser. ARES 2018. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3230833.3234688 [zugegriffen am: 14.02.2021]

[47] "APTSimulator." [Online]. Available: https://github.com/NextronSystems/APTSimulator [zugegriffen am: 14.01.2021]