HAW HAMBURG

Masterarbeit

Fabian Reiber

Exemplarische Erzeugung von Cyber Threat Intelligence (CTI) aus einer Exploit-Analyse und deren Integration in die Malware Information Sharing Platform (MISP)

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Fabian Reiber

Exemplarische Erzeugung von Cyber Threat Intelligence (CTI) aus einer Exploit-Analyse und deren Integration in die Malware Information Sharing Platform (MISP)

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr. Bettina Buth

Eingereicht am: 31. Januar 2020

Fabian Reiber

Thema der Arbeit

Exemplarische Erzeugung von Cyber Threat Intelligence (CTI) aus einer Exploit-Analyse und deren Integration in die Malware Information Sharing Platform (MISP)

Stichworte

Cyber Threat Intelligence (CTI), Malware Information Sharing Platform (MISP), Exploit, Exploit Database, Python, Incident Response

Kurzzusammenfassung

Das Ziel der Arbeit ist die Vorstellung eines entwickelten Prototyps zur automatisierten Erzeugung von Cyber Threat Intelligence (CTI) durch eine Exploit-Analyse und ihre Integration in die Malware Information Sharing Platform (MISP). Zunächst ist der Begriff CTI detailliert beschrieben und zeigt weitere Kategorien davon auf. Weiterhin ist eine Vorstellung von MISP für den direkten Austausch gewonnener Analyseergebnisse gegeben. Überdies sind die Zielgruppen von CTI ausführlich beschrieben. Im Kern geht die Arbeit der Frage nach, ob Indikatoren zur Identifizierung von Sicherheitsvorfällen aus der Analyse von Exploits gewonnen werden können. Wenn dies der Falls ist, sind sie als CTI anzusehen, sodass die verschiedenen Zielgruppen davon profitieren können. Außerdem stellt die Vorstellung eines Anwendungsfalls einen praktischen Nutzen einiger gewonnener Ergebnisse vor. Dabei wird gezeigt, ob das Advisory-System des DFN-CERT Inkonsistenzen bzgl. der Bewertung von Schwachstellen und Advisories aufweist. Zur Beantwortung dieser und weiterer Fragen, ist der Entwurf und die Entwicklung eines Prototyps genau aufgeführt. Die Entwicklung des Prototyps hat gezeigt, dass eine Exploit-Analyse möglich ist und verschiedene Indikatoren gewonnen werden konnten. Es wurde deutlich, dass dies ca. 30% aller extrahierten Ergebnisse sind. Durch ihre Einsortierung in die Kategorien Technical und Tactical Threat Intelligence, können alle Zielgruppen davon profitieren. Die Analyse des Anwendungsfalls zeigt zusätzlich, dass das Advisory-System verschiedene Inkonsistenzen in der Bewertung aufweist. Letztlich zeigt sich insgesamt ein steigendes Interesse an der Verwendung von CTI, sodass die Weiterentwicklung und Optimierung dieses Prototyps sinnvoll ist.

Fabian Reiber

Title of Thesis

Exemplary creation of Cyber Threat Intelligence (CTI) from an exploit analysis and its integration into the Malware Information Sharing Platform (MISP)

Keywords

Cyber Threat Intelligence (CTI), Malware Information Sharing Platform (MISP), Exploit, Exploit Database, Python, Incident Response

Abstract

The aim of the thesis is to present the development of a prototype for the automated creation of Cyber Threat Intelligence (CTI) through an exploit analysis and its integration into the Malware Information Sharing Platform (MISP). First, the term CTI is described in detail and shows more categories thereof. Furthermore, MISP is presented as a direct exchange tool of analytical results. Moreover, the target groups of CTI are described in detail. In essence, the work addresses the question of wether indicators for identifying security incidents can be obtained from the analyses of exploits. If this is the case, they should be considered as CTI, so that the different target groups have a benefit. Moreover, a use case presents a practical benefit of some results. It shows wether the advisory system of DFN-CERT has inconsistencies in terms of the evaluation of vulnerabilities and advisories. To answer these and other questions, the design and development of a prototype is described in detail. The development of the prototype has shown that the exploit analysis is possible and various indicators could be obtained. It became clear that this is about 30% of all extracted results. By sorting them into the categories Technical and Tactical Threat Intelligence, all target groups profit. The use case analysis also shows that the advisory system has various inconsistencies in the evaluation. Finally, there is a growing interest in the use of CTI, so that a further development and optimization of this prototype is reasonable.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AST Abstract Syntax Tree.

BSI Bundesamt für Sicherheit in der Informationstechnik.

C&C Command-and-Control.

CERT Computer Emergency Response Team.

CISO Chief Information Security Officer.

CSIRT Computer Security Incident Response Team.

CTI Cyber Threat Intelligence.

CVE Common Vulnerabilities and Exposures.

CVSS Common Vulnerability Scoring System.

DoS Denial of Service.

EMEA Europe, Middle East and Africa.

ENISA European Union Agency for Cybersecurity.

FIRST Forum for Incident Response and Security Teams.

IODEF Incident Object Description Exchange Format.

IR Incident Response.

IRT Incident Response Team.

MISP Malware Information Sharing Platform.

NIST National Institute of Standards and Technology.

NVD National Vulnerability Database.

OpenIOC Open Indicator of Compromise.

PoC Proof of Concept.

RFC Requests for Comments.

RID Real-time Inter-network Defense.

SDO STIX Domain Object.

SIEM Security Information and Event Management.

SOC Security Operations Center.

SRO STIX Relationship Object.

STIX Structured Threat Information eXpression.

TAXII Trusted Automated eXchange of Indicator Information.

TI Threat Intelligence.

TIP Threat Intelligence Platform.

TTPs Tactics, Techniques and Procedures.

UUID Universally Unique Identifier.

Codeausschnitte

1 Einleitung

1.1 Motivation

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) berichtete in seiner aktuellen Publikation „Die Lage der IT-Sicherheit in Deutschland 2018“ [3] über die Entdeckung von ca. 390.000 neuer Computer-Malware pro Tag. Im Jahr zuvor lag dieser Wert bei ca. 280.000 pro Tag. Die Gesamtanzahl bekannter Malware lag bis Ende Mai 2018 bei ca. 800 Millionen. Laut dem BSI ist Malware die englische Bezeichnung für Schadprogramme. Sie haben das Ziel bösartige und schädliche Funktionen auf einem Computer auszuführen. Zum Beispiel ist ein Exploit eine Malware, welche Schwachstellen in Computerprogramme ausnutzt [3]. Aus diesem Grund ist jede Malware eine potentielle Bedrohung für jede Person, Behörde sowie jedes Unternehmen. Jegliche Informationen, die aus einer Malware gewonnen werden können, sind hilfreich für den aktuellen und zukünftigen Schutz vor gleiche oder ähnliche Malware. Cyber Threat Intelligence (CTI) befasst sich u.a. damit, diese Informationen zu sammeln und sie zwischen verschiedene Interessengruppen auszutauschen. Zur Sammlung, Darstellung und zum Austausch von CTI gibt es verschiedene Softwarelösungen. Eine derzeit viel verwendete Software ist die Malware Information Sharing Platform (MISP) [96].

Laut der Umfrage „Third Annual Study on Exchanging Cyber Threat Intelligence: There Has to Be a Better Way“ [73] aus dem Jahr 2017 wird CTI bereits zu 60% von 1200 befragten Unternehmen aus den USA und dem Wirtschaftsraum Europe, Middle East and Africa (EMEA)[26] genutzt. Im Jahr 2015 waren es 48% [73]. Ferner ist aus der gleichen Umfrage für das Jahr 2017 ersichtlich, dass 54% der befragten Unternehmen diese Informationen zudem austauschen. 2015 lag der Wert noch bei 40% [73]. Wie Pace u. a. [72] und Shackleford [84] zu entnehmen sind, werden die Informationen u.a. von Incident Response Teams (IRTs) genutzt, um Betroffene vor Bedrohungen schützen zu können. Ein IRT kann Bestandteil eines Computer Emergency Response Teams (CERTs) sein und ist für eine schnelle Erkennung von Vorfällen sowie deren Beseitigung zuständig.

Ein CERT hingegen bietet weitere Dienstleistungen an, wie z.B. die Sensibilisierung und Weiterbildung zu Themen der IT-Sicherheit [13]. Das BSI hat zudem ein Mobile Incident Response Team (MIRT) aufgebaut, um Betroffene direkt vor Ort unterstützen zu können [3]. Ein von einem Sicherheitsvorfall betroffenes Unternehmen, kann das BSI aufsuchen und um Unterstützung zur Wiederherstellung der Funktionsfähigkeit angegriffener Systeme bitten. Die in CTI enthaltenen Informationen kommen aus verschiedenen Quellen wie z.B. von CERTs, aus öffentlichen CTI-Feeds oder von kommerziellen Dienstleistern [84].

Die Verarbeitung, Bereitstellung sowie der Austausch von CTI erfährt durch den kontinuierlichen Anstieg von Malware und den damit einhergehenden Sicherheitsvorfällen an stetiger Bedeutung. Gleichzeitig ist zu erkennen, dass zusätzliche Strukturen zur Erkennung von Bedrohungen und der Bekämpfung konkreter Vorfälle in Unternehmen und Behörden aufgebaut werden. Vor diesem Hintergrund soll in dieser Arbeit eine prototypische Software zur Erstellung von CTI beschrieben werden. Hierbei werden durch eine Exploit-Analyse Informationen generiert und in die Malware Information Sharing Platform (MISP) exportiert. Diese können sodann zur Analyse von konkreten Vorfällen genutzt werden. Eine detailliertere Zielsetzung folgt im anschließenden Abschnitt.

1.2 Zielsetzung

Es gibt bereits verschiedene Quellen von Bedrohungsinformationen die in MISP einfließen [35]. Eine Quelle für analysierte Exploits ist derzeit nicht bekannt. Daher soll mit dieser Arbeit gezeigt werden, welche Informationen aus einer Exploit-Analyse gewonnen werden können. Hierfür soll ein Konzept und die Entwicklung einer Software zur automatisierten Analyse von öffentlich zugänglichen Exploits vorgestellt werden. Zunächst soll eine detailliertere Analyse zu der Fragestellung erfolgen. Anschließend sollen die relevanten Anforderungen an die Software aufgezeigt werden. Daran anknüpfend soll das Design und die konkrete Umsetzung vorgestellt werden. Das Ziel der Analyse soll das Gewinnen von Informationen sein, die zur Erstellung neuer oder Anreicherung vorliegender CTI dienen. Konkret sollen die Analyseergebnisse nach MISP exportiert werden. MISP dient auf der einen Seite zur Darstellung sowie zum Austausch von Bedrohungsinformationen. Der Austausch erfolgt zwischen verschiedenen miteinander verbundenen MISP-Instanzen anderer Organisationen [96]. Auf der anderen Seite korreliert die Plattform neue Bedrohungsinformationen mit bereits existierenden. Dieses Wissen kann beispielsweise den

Analyst*innen eines im Unternehmen angesiedelten IRT zur Abschätzung potentieller Bedrohungen des Unternehmens dienen.

Ein weiteres Ziel soll die Generierung von Statistiken über die gewonnenen Ergebnisse sein. Die Exploits sollen aus der Exploit Database (ExploitDB) von Offensive Security bezogen werden und besitzen bereits diverse Metadaten [68]. Mit deren Hilfe und den analysierten Ergebnissen sollen verschiedene Trends der letzten Jahre über die Entwicklung und Veröffentlichung der Exploits gewonnen werden. Daraus soll z.B. erkannt werden, ob die Anzahl der Exploits mit der Anzahl dokumentierter Schwachstellen in Verbindung steht. Oder wie sich die Verteilung verwendeter Programmiersprachen zur Erstellung von Exploits verhält.

Ferner soll eine Einordnung der erzeugten Bedrohungsinformationen in die verschiedenen Informationskategorien von CTI vorgenommen werden. Zudem sollen mögliche Zielgruppen der erzeugten Analyseergebnisse benannt werden. In einem weiteren Abschnitt sollen die Grenzen der Exploit-Analyse dargestellt und diskutiert werden. Zuletzt soll ein konkreter Anwendungsfall vorgestellt werden. Darin wird das Advisory-System (ADV-System) des DFN-CERT vorgestellt und die darin enthaltenden Bewertungen von Schwachstellen und Advisories auf ihre Konsistenz hin überprüft.

Aus den vorangegangenen Fragestellungen ergibt sich die Notwendigkeit zur detaillierten Vorstellung des Begriffes Cyber Threat Intelligence, möglicher Formate zur Repräsentation sowie Wege des Austauschs dieser Informationen. Diese und die Vorstellung der Zielgruppen von CTI sollen die Grundlagen dieser Arbeit darstellen.

1.3 Kapitelübersicht

Nach der Motivation und Zielsetzung in Kapitel 1 werden in Kapitel 2 zwei Definitionen von CTI ausführlich vorgestellt und miteinander verglichen sowie die Vorteile und Herausforderungen durch den Einsatz von CTI diskutiert. Zusätzlich wird ein CTI-Lifecycle erläutert und verschiedene Datenformate sowie Protokolle für den Austausch von CTI aufgeführt. Eine Vorstellung von MISP und der ExploitDB schließen dieses Kapitel ab.

In Kapitel 3 werden diverse Zielgruppen von CTI vorgestellt. Diese setzen die erlangten Informationen auf unterschiedliche Weise ein und verfolgen verschiedene Ziele damit.

Anschließend werden die Forschungsfragen und dazugehörige Hypothesen an die zu entwickelnde Software zur Exploit-Analyse in Kapitel 4 aufgestellt. Außerdem werden die funktionalen und nicht-funktionalen Anforderungen an die Software aufgelistet.

Das 5. Kapitel stellt zunächst das Software-Design vor. Nachfolgend wird die konkrete technische Realisierung der Software aufgezeigt.

In der Evaluation in Kapitel 6 werden die gestellten Forschungsfragen beantwortet und die dazugehörigen Hypothesen bestätigt oder widerlegt. Außerdem werden diverse statistische Auswertungen gezeigt und diskutiert. Zudem werden verschiedene Hindernisse die während der Entwicklung aufgetreten sind aufgezeigt. Ferner wird in einem konkreten Anwendungsfall die Verwendung der Software eingeordnet.

Zuletzt wird die Arbeit mit einem Fazit sowie einem Ausblick in Kapitel 7 abgeschlossen.

2 Cyber Threat Intelligence

In diesem Kapitel werden zunächst zwei Definitionen zu CTI detailliert vorgestellt. Darauf aufbauend wird aufgezeigt, an welcher Definition sich der weitere Inhalt dieser Arbeit orientiert. Weiterhin werden diverse Vorteile und Herausforderungen durch den Einsatz von CTI erörtert. Anschließend wird ein CTI-Lifecycle aufgezeigt, anhand dessen eine Einordnung der Exploit-Analysesoftware stattfinden wird. Nachfolgend werden einige populäre sowie standardisierte Datenformate und Protokolle für den Austausch von CTI genauer beleuchtet. Im abschließenden Abschnitt 2.4 wird MISP als Austausch von CTI sowie die ExploitDB von Offensive Security präsentiert.

2.1 Definition

Innerhalb dieses Abschnittes werden zwei Definitionen von CTI vorgestellt. Die Darstellung unterschiedlicher Definitionen soll verdeutlichen, inwiefern sie sich unterscheiden. Letztlich soll eine passende Definition gewählt werden, an die sich die Inhalte nachfolgender Kapitel orientieren.

CTI nach Chismon und Ruks [4]

Chismon und Ruks [4] definieren zunächst den Begriff *Intelligence*. Intelligence sind Informationen die eingesetzt werden, um auf bestimmte Ereignisse reagieren zu können. Im Kontext von Threat Intelligence (TI) sind Bedrohungen (Threats) die Ereignisse. Entsprechend wird Threat Intelligence zur Unterstützung von Entscheidungen eingesetzt, um sich vor möglichen Angriffen zu schützen und Risiken besser bewerten zu können. Ferner gibt es drei unterschiedliche Bereiche in die Bedrohungen und Risiken kategorisiert werden können. Die Bereiche hängen vom Umfang der Informationen die zu einer Bedrohung oder einem Risiko bekannt sind ab und werden in Tabelle 2.1 kurz beschrieben.

Bezeichnung	Beschreibung
unknown unknowns	Bedrohungen und Risiken dieser Kategorie sollten möglichst nicht vorkommen, da ihre Existenz vollkommen unbekannt ist.
known unknowns	Bedrohungen und Risiken dieser Kategorie sind bekannt, allerdings fehlt es an dem Wissen über das Wie, Wo, Wann, Wer oder Warum.
known knowns	Bedrohungen und Risiken dieser Kategorie sind bekannt, konnten nachvollzogen und somit reduziert oder abgeschwächt werden. Eine Reduzierung bzw. Abschwächung konnte durchgeführt werden, da im Gegensatz zu *known unknowns* mehr Wissen über die Bedrohungen und den Risiken vorlag.

Tabelle 2.1
Bereiche zur Kategorisierung von Bedrohungen und Risiken

Mit jedem Wechsel der Kategorie, von *unknown unknowns* zu *known unknowns* sowie von *known unknowns* zu *known knowns*, wird der Informationsumfang zu jeder Bedrohung und jedem Risiko größer. Daher muss versucht werden, Bedrohungen und Risiken aus den beiden zuerst genannten Kategorien in die zuletzt genannte Kategorie zu verschieben. [4]

Zum Schutz vor Bedrohungen werden viele verschiedene Quellen herangezogen aus denen Bedrohungsinformationen gesammelt werden. Die Informationen sind für verschiedene Zielgruppen von Bedeutung und vermitteln unterschiedliches Wissen. Für eine konkretere Definition der Informationen und der Zielgruppen sowie der Abgrenzung zueinander, wird Threat Intelligence in vier Kategorien eingeteilt. Alle Kategorien unterscheiden sich zudem in den Punkten Analyse, Austausch sowie Verwendung. Diese Unterteilung wurde von Chismon und Ruks [4] vorgenommen und wird im Folgenden genauer erläutert.[1]

Strategic Threat Intelligence spricht Entscheidungsträger*innen an und sind weniger technisch beschrieben. Die Erkennung von Bedrohungen auf strategischer Ebene führen zu den Zielen neue Risiken zu erkennen und aktuelle besser zu verstehen. Mit den Informationen werden Entscheidungen getroffen und ihre Auswirkungen im Unternehmen nachvollzogen. Gerade die finanziellen Auswirkungen stehen dabei im Fokus. Sie werden in Berichte

[1] Die Reihenfolge der Auflistung stellt keine Wichtigkeit für die Zielgruppen dar.

verfasst oder bei Briefings erläutert. Eine Analyse der Informationen kann sehr komplex werden. Daher richten sich die Entscheidungsträger*innen nach Trends, Beobachtungen oder Intentionen. Zudem wird eine hohe Expertise, vor allem in sozialpolitischen Analysen, verlangt, sodass einige Entscheidungsträger*innen externe Expert*innen beauftragen die bei der Analyse unterstützen. Bei der Evaluation von Strategic TI kommt es auf die Korrektheit, Benutzbarkeit und Aktualität der Informationen an. Strategic TI werden selten ausgetauscht, da sie sensible Informationen über das Unternehmen beinhalten können. Der Austausch sollte gut durchdacht sein. Daher wird geraten, andere Arten von Threat Intelligence zu teilen, damit sie andere Entscheidungsträger*innen im strategischen Bereich unterstützen können. [4]

Operational Threat Intelligence sind sehr konkrete Informationen über bevorstehende Angriffe. Sie richten sich an Sicherheitsbeauftragende oder an die Leitung eines IRTs. Durch die vorhandenen Informationen soll eine Abmilderung des Angriffs erreicht werden. Operational TI sind sehr informativ, da sie Hinweise auf konkrete Fragestellungen geben, wie z.B. wer greift wann in welcher Form an. Sie sind allerdings sehr selten und nur schwer für die Zielgruppe zu erhalten, da die Angreifenden dafür überwacht werden müssten. Deshalb sollte sich ein Unternehmen auf legale Art und Weise diese Art der Bedrohungsinformationen beschaffen, wie z.B. durch Überwachung öffentlicher Foren oder den sozialen Medien. Angriffe können während der Analyse mit realen Ereignissen, die einen Zusammenhang mit dem eigenen oder anderen Unternehmen aufweisen, in Verbindung gebracht werden. Daraus können Muster herausgefiltert werden, die zur Abwehr zukünftiger Angriffe hilfreich sein können. Soziale Medien oder öffentliche Foren bieten hingegen sehr umfangreiche und fehlerhafte Informationen. Um verwertbare Informationen daraus zu erhalten, ist es deshalb sinnvoll sie automatisiert auf Muster zu analysieren. Eine vorherige manuelle Analyse ist für diese automatisierte Mustererkennung zudem nicht zu ignorieren. Eine Kombination von Operational und Tactical Threat Intelligence bspw. kann zu genaueren Erkenntnissen eines Angriffs führen. Operational TI kann als erfolgreich bewertet werden, sobald ein Angriff dadurch prognostiziert sowie abgeschwächt bzw. abgewehrt wurde. In den meisten Fällen ist es für private Unternehmen unwahrscheinlich diese Art von TI zu erhalten. Dadurch ist eine Analyse der Ursache nach dem Angriff zielführender für zukünftige Optimierungen. Ebenso sollte in andere Arten von Threat Intelligence investiert werden. Ein Austausch dieser Informationen kann zur Abwehr von Angriffen auf andere Unternehmen hilfreich sein. [4]

Tactical Threat Intelligence sind sehr hilfreiche Informationen über die Taktiken, eingesetzten Werkzeuge und Methoden der Angreifenden. Sie werden auch als *Tactics, Techni-*

ques and Procedures (TTPs) bezeichnet. Sie geben u.a. Administrator*innen oder einem IRT Aufschluss wie ein Angriff durchgeführt wurde. Das Ziel ist die Sicherstellung der Funktionalität eingesetzter Mittel zur Abwehr von Angriffen. Tactical TI können z.B. aus Berichten verschiedener Tactical TI Anbieter oder durch den Austausch mit anderen Organisationen bezogen werden. Das Ziel der Analyse besteht darin, Indikatoren über einen Angriff herauszufiltern, um die Vorgehensweise der Angreifenden zu verstehen. Im Idealfall werden Informationen zum Angriffsmuster erkannt, welche auch in Zukunft zum Schutz gegen Angriffe genutzt werden können. Dafür werden die eingesetzten Werkzeuge sowie die Kommunikation der infizierten Geräte mit den Angreifenden genau analysiert. Ebenso ist es hilfreich zu erkennen, wie die Angreifenden sich selbst vor der Erkennung schützen. Während der Evaluation von Tactical TI muss geklärt werden, ob diese zu einer erfolgreichen Abwehr von Angriffen beigetragen haben. Konnte der Angriff dennoch erfolgreich durchgeführt werden, muss dieser genau untersucht werden und führt somit zu einem Analyseprozess. Der Austausch von Tactical TI trägt dazu bei, dass sich andere Unternehmen oder Organisationen vor ähnlichen oder gleichen Angriffen schützen können. Idealerweise werden die Bedrohungsinformationen in einem Incident Report strukturiert mit allen Informationen über die TTPs zusammengefasst. [4]

Technical Threat Intelligence sind technische Informationen zu einem Angriff. Sie sind im Gegensatz zu Tactical Threat Intelligence konkrete Indikatoren und sehr viel kurzlebiger. Beispielsweise fällt eine spezielle Malware unter Tactical TI und eine IP-Adresse eines Command-and-Control (C&C)-Servers unter Technical TI. Die Informationen sind für technische Systeme vorgesehen, wie bspw. Firewalls zur Blockierung von Verbindungen zu C&C-Servern oder zur Filterung von infizierten E-Mails. Dies kann zu einer sehr frühen Erkennung und Blockierung von Angriffen führen. Eine Herausforderung sind die großen Datenmengen die durch Technical TI zustande kommen. Daher ist es entscheidend, welche Informationen gesammelt und analysiert werden. Der Nutzen dieser Informationen wird zudem kritisch gesehen. Zum Beispiel kann der Bezug großer Datenmengen der verschiedenen Anbieter von Technical TI zu einem Informationsverlust führen, sodass sie für die Analyse unbrauchbar werden. Die Verarbeitung und Analyse von Technical TI sollte automatisiert durchgeführt werden, da ihre Lebenszeit sehr gering ist und sie sich kontinuierlich ändern. Durch ihre Darstellung in verschiedene Datenformate wie z.B. STIX oder OpenIOC, bietet sich zudem die maschinelle Verarbeitung an. Zur Visualisierung, Einsichtnahme und manuellen Bearbeitung von Technical TI wurden in den letzten Jahren diverse Werkzeuge entwickelt. Es ist schwierig eine Bewertung von Technical TI und Analysewerkzeugen im Sinne ihrer Qualität durchzuführen, da nicht gewährleistet werden

kann, ob sie wirklich zu einem Schutz beitragen können. Es ist nicht sichergestellt, ob mit jenen Technical TI oder Analysewerkzeugen die gleiche Anzahl von Angriffen verhindert wird, als mit anderen. Zudem kann eine Fixierung ausschließlich auf diese Art der Informationen sehr ineffizient sein, da die Angreifenden problemlos individuellere Angriffe durchführen können. Eine ausgewogene Balance der Nutzung verschiedener TI ist daher ratsam. Der Austausch von Technical TI sollte in jedem Fall erfolgen. Durch einheitliche Datenformate ist zudem ein einfacher Austausch zwischen privaten Unternehmen und Organisationen gewährleistet. [4]

CTI nach Johnson u. a. [27]

Das National Institute of Standards and Technology (NIST) nutzt in seiner Veröffentlichung den Begriff *Threat Information*. In diesem Kontext ist Threat regelmäßig nur als Cyber Threat zu verstehen, also jegliche Informationen zu einer digitalen Bedrohung, die einer Organisation helfen, sich vor digitalen Vorfällen zu schützen. Weiterhin sollen sie die Identifizierung der Aktivitäten eines Angreifenden unterstützen. Zudem wird in der Publikation der Nutzen und die Herausforderungen des Austausches von Threat Information vorgestellt. In den nächsten Abschnitten werden die fünf vom NIST definierten Arten von Bedrohungsinformationen vorgestellt. [27]

Indicators sind technische Informationen oder Ereignisse die auf einen drohenden oder bereits durchgeführten Angriff hinweisen. Sie können dazu genutzt werden potentielle Bedrohungen zu erkennen oder sich davor zu schützen. Die IP-Adresse eines verdächtigen C&C-Servers ist ein Beispiel für einen Indicator. Wurde ein System bereits durch eine Malware kompromittiert, kann sich diese mit einem C&C-Server verbinden, um weitere Malware nachzuladen oder Daten zu entwenden. Die IP-Adresse kann sodann zur Konfiguration von Firewallregeln eingesetzt werden, sodass zukünftige Verbindungen zu dem Server unterbunden werden. [27]

Tactics, Techniques and Procedures (TTPs) geben Aufschluss über das Verhalten eines Angreifenden. Tactics sind *high-level* Informationen zur Beschreibung des Verhaltens. Techniques hingegen definieren das Verhalten detaillierter. Procedures sind *low-level* Informationen und beschreiben die Techniques-Informationen weiter im Detail. Durch diese Kette von Informationen kann die Neigung eines Angreifenden z.B. zu einer präferierten Malware oder eines Angriffswerkzeugs verdeutlicht werden. [27]

Durch *Security Alerts* werden Schwachstellen, Exploits oder andere Sicherheitsrelevanten technischen Details kurz und menschenlesbar zusammengefasst. Sie werden u.a. von CERTs, kommerziellen Dienstleistungsunternehmen im Bereich IT-Sicherheit oder Forscher*innen bereitgestellt. [27]

Threat Intelligence Reports fassen die gesammelten Informationen zu einer Bedrohung, wie z.B. TTPs oder Informationen über das anzugreifende Ziel, zusammen. Sie geben übergeordnete Erkenntnisse über die Bedrohung. In diesen Berichten werden die jeweiligen Threat Information aggregiert, analysiert und interpretiert zusammengefasst, sodass die Entscheidungsträger*innen eines Unternehmens gezielt handeln können. [27]

Tool Configurations beschreiben Empfehlungen über einzusetzende Werkzeuge und wie sie zu konfigurieren sind, damit Bedrohungsinformationen automatisiert gesammelt, verarbeitet und analysiert werden. Dazu zählen u.a. Regeln für eine Firewall oder die Konfiguration von Webfiltern. [27]

Johnson u. a. [27] differenzieren zwischen dem Austausch von Bedrohungsinformationen innerhalb einer Organisation und über die Organisationsgrenzen hinaus. Innerhalb der Organisation bspw. wird zunächst ein kompromittiertes System erkannt. Dieser Vorfall wird mit diversen Indikatoren zusammengefasst und mit Systemadministrator*innen ausgetauscht. Im Anschluss wird Software zur Erkennung von Angriffen auf anderen Systemen konfiguriert, damit dieser Vorfall in Zukunft direkt erkannt und abgewehrt wird. [27]

Vergleich der vorgestellten Definitionen zu CTI

Die Definition nach Chismon und Ruks [4] fällt deutlich ausführlicher aus und beschreibt die einzelnen Typen von Bedrohungsinformationen abstrakter sowie allgemeiner als die Definition nach Johnson u. a. [27]. Die Benennung der Typen aus Johnson u. a. [27] sind zum Teil in den allgemeiner gefassten Typen von Chismon und Ruks [4] enthalten. Die insgesamte Bedeutung von CTI wird somit in beiden Quellen sehr ähnlich bewertet.

Die nachfolgenden Kapitel orientieren sich nach der Definition von Chismon und Ruks [4], da diese das Einsortieren zusätzlicher Begriffe sowie das Aufzeigen weiterer Argumente deutlicher und einfacher gestaltet.

Vorteile durch den Einsatz von CTI

Shackleford [84] vergleicht in einer Studie aus dem Jahr 2018 diverse Umfragen zum Thema CTI aus den letzten drei Jahren. Aus der Einschätzung der Befragten geht hervor, dass detaillierte Indikatoren von eingesetzter Malware am wertvollsten für Unternehmen sind. Knapp darauf folgen Informationen über Schwachstellen die gezielt von Angreifenden ausgenutzt werden. Zudem werden in der Studie Antworten von befragten Unternehmen zitiert. Aus ihnen ist ersichtlich, wie sie die Indikatoren für einen besseren Schutz einsetzen und welche Vorteile sie bieten. Gibt es zum Beispiel neue Informationen zu einer Schwachstelle sowie die Information, dass diese bereits ausgenutzt wird, kann vorausschauend darauf reagiert werden.

Weiterhin wird die Zufriedenheit von CTI aus verschiedenen Blickwinkeln betrachtet. 76% der Befragten gaben an, sie seien zufrieden mit der Verwendung von Bedrohungsinformationen, um darin Suchen oder sie für Berichte verwenden zu können. Knapp 71% verwenden die Informationen für Berichte auf strategischer sowie operativer Ebene. Genau so viele finden sie brauchbar, um Bedrohungen und Indikatoren sichtbar zu machen. Diese Werte geben einen kleinen Einblick, wie sinnvoll Bedrohungsinformationen mittlerweile sind. [84]

2016 und 2017 gaben 64% bzw. 78% der Befragten an, CTI verbessere den Schutz, die Erkennung sowie die Reaktion auf Bedrohungen. 2018 erhöhte sich der Prozentsatz um weitere 3% auf 81%. Aus diesem Trend schließt Shackleford [84] die allgemeine Aussage, CTI sei hilfreich zur Behandlung von Bedrohungen. Ferner wird in der Studie erörtert, wie die Verwendung von Bedrohungsinformationen die Sicherheit verbessert hat. An erster Stelle mit etwas mehr als 70% wurde 2018 ausgesagt, dass CTI die Sichtbarkeit von Bedrohungen und Angriffsmethoden, die auf das Unternehmen einwirken können, verbessert wurde. Darauf folgt mit 70% die Verbesserung von Sicherheitsoperationen. Diese umfassen u.a. das Blocken von bösartigen Aktivitäten oder Bedrohungen. Zudem gaben 65% der Unternehmen an, dass CTI die Erkennung unbekannter Bedrohungen verbessere. Ob es sich um gänzlich unbekannte oder nur für die Unternehmen unbekannte Bedrohungen handelt, geht aus der Studie nicht hervor. [84]

Darüber hinaus erweist sich der Austausch von CTI zwischen verschiedenen Unternehmen als profitabel und sinnvoll im Hinblick auf etwaige Schutzmaßnahmen. Besonders Unternehmen die in der gleichen Branche tätigt sind und ggf. ähnliche oder gleiche Systeme

verwenden, genießen den Vorteil, dass ausgetauschte TTPs direkt genutzt werden können. Der Austausch trägt zur Reduzierung von Risiken bei und verbessert die allgemeine Sicherheit eines Unternehmens. [27]

Herausforderungen beim Einsatz von CTI

Neben den Vorteilen von CTI gibt es zudem diverse Herausforderungen und Barrieren für deren Einsatz. Der bedeutendste Faktor ist das Fehlen von erfahrenem Personal in den Unternehmen, um CTI effektiv einsetzen zu können. Laut der Studie hat sich dies im Jahr 2018 weiter verschärft. Demnach gaben 62% der Unternehmen an, dass dies die größte Hürde zur Verwendung von CTI sei. 2017 waren es 53%. Mit knapp 50% folgen darauf die Hindernisse, dass das notwendige Budget sowie die Zeit fehlen, neue Prozesse im Unternehmen zu etablieren. Im Vergleich zum Jahr 2017 haben sich diese Werte nicht geändert. [84]

Zudem betreffen die Herausforderungen sowohl das Konsumieren als auch das Erstellen von CTI. Eine grundlegende Hürde ist das Vertrauen zwischen den Parteien die Informationen austauschen. Eine regelmäßige Kommunikation zwischen den Parteien kann diese Hürde überwinden. Eine weitere Problematik stellt die Interoperabilität der Systeme und Automatisierung des Prozesses für den Austausch dar. Der Einsatz von Standards für das Datenformat und von Austauschprotokollen, erleichtert die Automatisierung. Dies bedingt allerdings den Einsatz von Zeit und Ressourcen, welche besonders hoch sein können, wenn unterschiedliche Formate und Protokolle zum Einsatz kommen. Für den Schutz sensitiver Informationen sollten zudem Schutzmechanismen eingerichtet werden. Bei Bekanntwerden sensibler Informationen kann ein finanzieller Verlust entstehen oder es können rechtliche Maßnahmen folgen. [27]

Eine weitere besondere Herausforderung, die beim Konsum von CTI auftritt, ist die Qualität der bezogenen Daten. Es muss gewährleistet sein, dass die Informationen korrekt und sachdienlich sind. Zugleich verbirgt die weitere Nutzung oder die Ignoranz der erhaltenen Informationen mögliche Auswirkungen die ein potentielles Risiko für das Unternehmen darstellen können. Die Bereitstellung von CTI steht u.a. vor der Hürde, dass die Informationen aus rechtlicher Sicht oder Datenschutzgründen gekürzt sein müssen. Folglich können diese Informationen weniger brauchbar oder qualitativ schlechter sein. [27]

2.2 Lifecycle

In Abbildung 2.1 ist der CTI-Lifecycle nach Pace u. a. [72] dargestellt und umfasst sechs Phasen. Chismon und Ruks [4] definieren ebenfalls einen Lifecycle. Dieser ist weitestgehend identisch sowie kompatibel mit diesem und wird daher nicht vorgestellt. Weiterhin ist in dem Schaubild ersichtlich welche Zielgruppen konkrete Bedrohungsinformationen, die durch den Lifecycle entstehen, erhalten können. Diese und weitere Zielgruppen werden in Kapitel 3 umfassender aufgezeigt. Im Folgenden werden die jeweiligen Phasen genauer erörtert.

Die *Direction*-Phase bündelt die Ziele und Anforderungen an eine CTI-Lösung. Es sollte klar definiert sein welche Bereiche mit welcher Priorität in einem Unternehmen zu schützen sind. Ferner muss klar sein, welche Threat Intelligence notwendig sind, um die Ziele erreichen zu können. Die Auswirkungen von möglichen Angriffen ist ein weiterer Punkt der in dieser Phase heraus gearbeitet werden muss. Eine ausgearbeitete Liste von Zielen und ihren Anforderungen wurde von Pace u. a. [72] ausgearbeitet und ist in den Appendix ihrer Ausarbeitung zu finden.

Zur Erfüllung der ausgearbeiteten Ziele und Anforderungen, müssen in der *Collection*-Phase Daten gesammelt werden. Die Daten sollten aus verschiedenen Quellen stammen. Abbildung 2.1 zeigt drei verschiedene Kategorien, aus denen die Daten bezogen werden können. *Internal Sources* sind Daten die im eigenen Unternehmen gesammelt wurden, wie z.B. Firewall Logs oder aufgezeichnete Netzwerkpakete. *Technical Sources* sind externe Daten aus Threat Feeds oder Schwachstellendatenbanken. Als weitere Quelle wird *Human Sources* genannt. Diese umfassen soziale Medien oder Foren im Dark- oder Clearweb. Ein zusätzlicher nicht zu ignorierender Punkt ist die Automatisierung dieser Phase. Analyst*innen sollten möglichst viel Zeit für die manuelle Analyse der Bedrohungen haben. [72]

Die Verwendung unterschiedlicher Quellen führt dazu, dass die Daten in unterschiedlichen Formaten vorliegen. Daher sollten sie in der *Processing*-Phase vereinheitlicht werden. Zusätzlich kann dies bedeuten, dass die Daten nicht 1:1 transformiert werden können. Bei Bedarf müssen relevante Informationen aus den Daten extrahiert werden. Diese Phase sollte zudem möglichst automatisiert durchlaufen werden. [72]

Die Ergebnisse der *Analysis*-Phase müssen der Zielgruppe entsprechend verständlich und nachvollziehbar aufbereitet werden. Sind sie bspw. für den Vorstand vorgesehen, sollten

sie keine technischen Details enthalten, sehr kurz sein und Empfehlungen für das weitere Handeln enthalten. Konkrete technische Informationen die analysiert wurden, haben eine andere Zielgruppe und bedürfen daher ein anderes Format. [72]

In der *Dissemination*-Phase werden die gewonnenen Erkenntnisse aus der Analyse an die jeweilige Zielgruppe weitergeleitet. Darüber hinaus sind diverse Fragen zu klären, wie u.a. in welchem Format die Ergebnisse ausgeliefert werden sollen, oder wie bei Rückfragen vorgegangen wird. [72]

Zuletzt wird in der *Feedback*-Phase reflektiert, ob die zuvor erstellten Anforderungen erfüllt wurden. Jede Zielgruppe gibt eine Rückmeldung zu den erfassten Informationen, sodass jede Phase nochmals genauer beleuchtet und hinterfragt werden kann. Wurden die Anforderungen erfüllt, können diese verfeinert oder ausgebaut werden. Konnten sie nicht erfüllt werden, muss festgestellt werden, in welcher Phase welche Änderungen durchgeführt werden müssen. Zur Rückmeldung können für jede Zielgruppe formale sowie nicht-formale Kommunikationskanäle aufgebaut werden. [72]

In Abschnitt 4.2 wird eine Eingliederung der hier vorgestellten Software zur Analyse von Exploits in den CTI-Lifecycle vorgenommen.

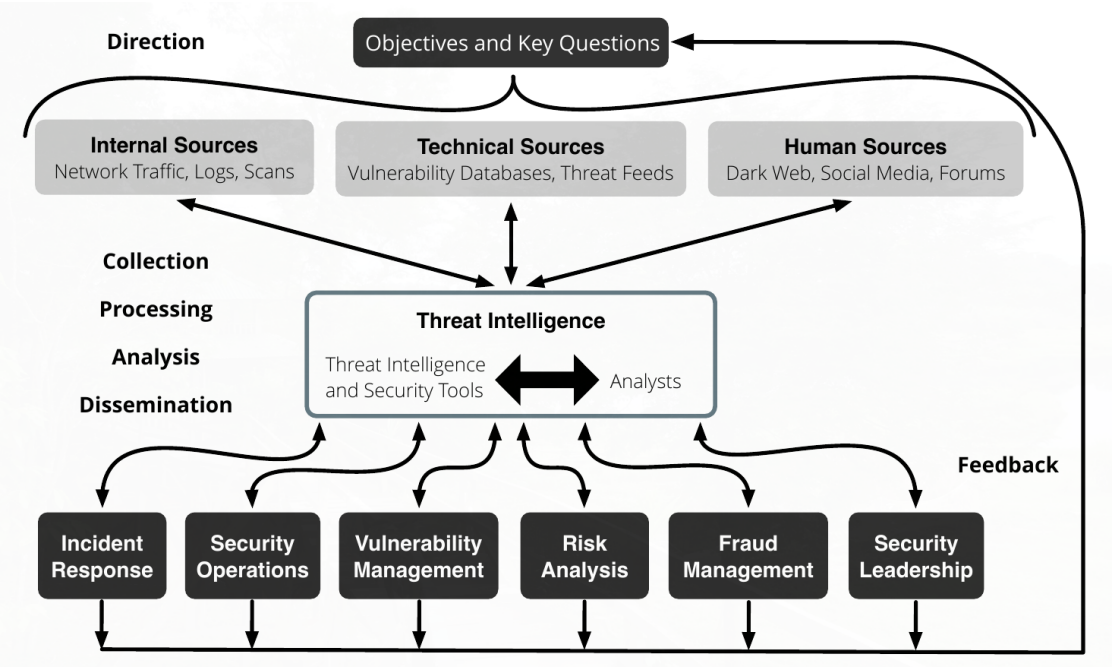

Abbildung 2.1
Sechs Phasen des CTI-Lifecycle [72]

2.3 Standards

Wie bereits in Kapitel 2.1 erwähnt, wird die Definition nach Chismon und Ruks [4] zum Verständnis von CTI in dieser Arbeit genutzt. Demzufolge sollten Technical Threat Intelligence automatisiert geteilt werden. Diese Automatisierung setzt zum einen geeignete Datenformate zur Repräsentation und zum anderen einheitliche Transportprotokolle für den Austausch der Informationen zwischen unterschiedlichen Parteien voraus. Dies sind zwei von drei wichtige Faktoren die Kampanakis [29] erörtert. Datenschutz ist der dritte Punkt, welcher hier nicht Gegenstand der Diskussion sein wird.

Aufgrund der beiden ersten Punkte haben sich in der Vergangenheit verschiedene Standards für unterschiedliche Anwendungsfälle entwickelt. Shackleford [83] erfasste, welche Standards für die Verarbeitung von CTI in Unternehmen eingesetzt werden. Die befragten Unternehmen haben ihre Niederlassungen mit mehr als 75% in den USA, 40% in Europa und mit 34% in Asien. Weitere Betriebe sind zudem in Kanada, Australien, Neuseeland, im mittleren Osten, Südamerika und Afrika angesiedelt. Der Studie ist zu entnehmen, dass Structured Threat Information eXpression (STIX) der am verbreitetste Standard für Datenformate ist, gefolgt von Open Indicator of Compromise (OpenIOC). MISP als Austauschplattform mit einem eigenen Kernformat und einem REST-API, werden von knapp 26% der Befragten eingesetzt. Incident Object Description Exchange Format (IODEF) hingegen wird von ca. 14%, der an der Umfrage beteiligten Unternehmen, genutzt. Eine separate Befragung zu den Austauschprotokollen ist in der Studie nicht aufgeführt. Dennoch kann nicht davon ausgegangen werden, dass Informationen die in STIX oder IODEF repräsentiert sind, mithilfe der Protokolle Trusted Automated eXchange of Indicator Information (TAXII) respektive Real-time Inter-network Defense (RID) ausgetauscht werden. Auch wenn diese füreinander konzipiert wurden, sind sie dennoch unabhängig voneinander.

Darüber hinaus gaben ca. 25% der befragten Unternehmen an, dass fehlende Interoperabilität ein Hindernis für den effektiven Einsatz von CTI sei. Daher ist es wünschenswert, dass die Plattformen für den Datenaustausch interoperabel sind und mit der Vielzahl von Formaten und Protokollen umgehen können. Dies ist aber nicht immer gegeben [79]. Im folgenden Unterkapitel werden STIX, IODEF sowie das Kernformat von MISP detaillierter vorgestellt, welche zur Beschreibung von Vorfällen, Angriffsmustern oder Malware eingesetzt werden [29]. Im daran anschließenden Unterkapitel werden TAXII und RID als bekannte Protokolle für den Austausch von CTI genauer beschrieben.

2.3.1 Formate

Structured Threat Information eXpression (STIX)

Sauerwein u. a. [80] haben festgestellt, dass STIX der de-facto Standard in den Plattformen zum Import und Export von Bedrohungsinformationen ist. STIX ist eine quelloffene Sprache zur Definition von CTI und deren Austausch. Die Ziele von STIX sind, ein besseres Verständnis von Angriffen zu erlangen sowie diese möglichst schnell und effizient zu unterbinden. Die Informationen werden in zwei Objekttypen unterteilt abgespeichert, in denen jedes Objekt mit weiteren Informationen mittels Attributen angereichert werden kann. Die STIX Domain Objects (SDOs) können mittels STIX Relationship Objects (SROs) zu komplexeren Repräsentationen von CTI miteinander verknüpft werden. Es gibt zwölf SDOs und zwei SROs. Einige SDOs ähneln sich und können in Kategorien gruppiert werden. Beispielsweise können *Attack Pattern*, *Malware* sowie *Tool* als TTPs betrachtet werden, da sie u.a. das Verhalten der Angreifenden beschreiben. Im folgenden Abschnitt werden die SDOs kurz angerissen und im Anschluss wird auf die SROs eingegangen. Die jeweiligen Attribute und spezifischen Beziehungen von SDOs und SROs können dem offiziellen Entwurf von STIX entnommen werden. [28]

Attack Pattern beschreiben die möglichen Versuche eines Angreifenden seine Ziele zu kompromittieren. Dadurch können Angriffe kategorisiert sowie generalisiert werden und bieten zudem detailliertere Informationen über die Art und Weise des Angriffs. *Spear phishing* ist ein Beispiel dafür. Dabei werden gut formulierte E-Mails mit Malware im Anhang oder einem Link zum Download einer Malware, an potenzielle Opfer verschickt. In STIX können Attack Pattern eine Beziehung zu weiteren SDOs aufweisen, um bspw. deutlich zu machen, welche Malware eingesetzt wurde. [28]

Eine *Campaign* fasst das Verhalten einer Gruppe von Angreifenden mit fester Zielsetzung zusammen. Sie agieren gegen verschiedene Ziele über einen definierten Zeitraum mit diversen Angriffen. Sie können zudem Teil der SDOs Intrusion Set und Threat Actor sein. Campaigns werden neben der Zielsetzung zudem über die auftretenden Vorfälle, ihren Zielen und den eingesetzten Mitteln beschrieben. [28]

Durch *Course of Action* wird auf bestehende Angriffe mittels aktives Handeln reagiert. Dies umfasst technische und automatisierte Prozesse wie das Patchen einer Schwachstelle. Außerdem können Handlungen auf höherer Ebene durchgeführt werden wie die Durchführung von Weiterbildungen für die Mitarbeitenden. Dieser SDO kann u.a. mit

Vulnerability oder Attack Pattern in Beziehung gebracht werden, um zu erkennen wie sie abgeschwächt werden können. [28]

Über das *Identity*-Objekt werden Individuen, Unternehmen oder Organisationen identifiziert. Diese können auch als Klassen gruppiert und damit verknüpft werden. Dem Objekt werden Informationen, wie z.B. der Kontakt, beigefügt. Dadurch kann u.a. die Quelle der erstellten CTI oder der Angreifende repräsentiert werden. [28]

Durch *Indicator*-Objekte werden Muster beschrieben, die auf bösartige oder verdächtige Aktivitäten hinweisen. Sie spiegeln letztlich die selben Informationen wider, die bereits in Kapitel 2.1 unter dem Begriff Indicator erläutert wurde. Das Muster wird mit der STIX Pattering Language beschrieben [8]. Weiterhin ist der Zeitraum der Gültigkeit und Brauchbarkeit des Indicators in dem Objekt hinterlegt. [28]

Ein *Intrusion Set* bündelt Wissen über das Verhalten einer Gruppe von Angreifenden sowie Annahmen ihrer eingesetzten Hilfsmittel. Einem Set können mehrere Campaigns oder Aktivitäten angehören, die über gemeinsame Attribute - die auf einen bekannten oder unbekannten Threat Actor hinweisen - miteinander verknüpft sind. [28]

Durch ein *Malware*-Objekt wird eine bösartige Software für den unerlaubten Zugriff in ein IT-System unter anderem durch eine ausführliche Beschreibung erläutert. Viren und Würmer zählen zu einer Malware, gefährden die Vertraulichkeit, Integrität und Verfügbarkeit der Daten und sollen von den Opfern unentdeckt bleiben. Beziehungen können zu Vulnerability und Identity hergestellt werden oder Verweisen auf andere Malware SDOs die eine Variante dieser Malware darstellen. [28]

Observed Data ist eine kontextfreie Information (Rohdatum) über beobachtete Aktivitäten im Netzwerk oder in IT-Systemen. Eine beobachtete IP-Adresse oder Netzwerkverbindung sind Beispiele dafür. Sie unterscheidet sich zu einem Indicator dadurch, dass sie keine weiteren Informationen enthalten, als einen Zeitraum und die darin gezählten Vorkommnisse der beobachteten Aktivität. Dadurch können Sighting-Beziehungen u.a. von Indicators hergestellt werden, damit eine konkrete Beobachtung dokumentiert werden kann. Infolgedessen kann der Zeitraum und die Regelmäßigkeit einer aktiven Malware festgestellt werden. [28]

Ein *Report*-Objekt bündelt CTI verschiedener SDOs wie Malware oder Threat Actor in einem Bericht mit weiterem Kontext und Details zusammen. Es beinhaltet eine Liste zu allen SDOs und SROs auf den der Report Bezug nimmt. Hiermit wird ein umfassender Bericht zur Veröffentlichung bereitgestellt. [28]

Threat Actors sind Gruppen, Organisationen oder Individuen die ein böswilliges Verhalten aufweisen. Dieser SDO ist kein Intrusion Set, kann aber einem, wie bereits beschrieben, angehören. Threat Actors können mit Hilfsmitteln gezielt Angriffe durchführen oder an Campaigns beteiligt sein. Sie zeichnen sich u.a. durch ihre Motive, Fähigkeiten und zurückliegenden Aktivitäten aus. [28]

In einem *Tool*-Objekt werden Informationen über eingesetzte Werkzeuge zur Durchführung von Angriffen beschrieben. Sie unterscheiden sich zu Malware darin, dass es frei verfügbare und legitime Software ist. Netzwerkscanner oder Remote-Access Werkzeuge zählen dazu. Das Wissen über den Einsatz solcher Werkzeuge kann Aufschluss über die Ausführung einer Campaign geben. Mittels dieser SDO werden die Eigenschaften der Werkzeuge aufgezeigt und lassen Behauptungen zu, wie ein Threat Actor sie während eines Angriffes verwendet. [28]

Vulnerability beschreiben Schwachstellen in einer Software die ausgenutzt werden können, um sich unberechtigten Zugriff zu einem System zu verschaffen. Sie können neben einer eigenen Beschreibung zudem externe Verweise, z.B. zu einem Eintrag in einer CVE-Datenbank, enthalten. [28]

Mittels der STIX Relationship Object (SRO) wird die Beziehung zwischen zwei SDOs beschrieben. Auf der einen Seite gibt es ein *Generic-SRO* mit verschiedenen Typen von Beziehungen. Auf der anderen Seite gibt es ein spezielleres Sighting SRO mit zusätzlichen Attributen zur genaueren Erläuterung von konkreten Beobachtungen von SDOs. Eine Beziehung zwischen zwei SDOs kann als Graph visualisiert werden. Die SDOs sind die Knoten und die SRO ist die Kante die die beiden Objekte verbindet. [28]

In der Dokumentation ist für jedes SDO eine Tabelle mit definierten Typen von Generic-SROs aufgeführt. Sie definiert zudem, welche Objekte mit welchen anderen Objekten miteinander in Beziehung gebracht werden sollten. Zur Wahrung der Konsistenz sollten sie wie angegeben benutzt werden. Allerdings ist es möglich jedes SDO mit jedem anderen über die *related-to* oder einer individuell benannten Beziehung zu verbinden. Abbildung 2.2 zeigt ein Beispiel in dem ein Indicator mit einem Malware-Objekt über eine *indicates*-Beziehung verknüpft ist. [28]

Sighting SROs geben Auskunft über etwas was gesehen wurde, ähnlich den Observed Data Objekten. Sie werden eingesetzt, um zu erkennen wer oder was Ziel eines Angriffs sowie wie dieser Angriff durchgeführt wurde und zeigt den Verlauf eines Angriffs auf. Die

zusätzlichen Eigenschaften in diesen Objekten sind nicht Teil der Generic-Beziehung, sondern repräsentieren eine spezifische Beobachtung die gezählt oder in einem Zeitintervall erfasst wurde. Der Grund für diese spezielle Form der Beziehung ist, dass Beobachtungen nicht existieren können, ohne eine Beziehung zu dem was gesichtet wurde zu haben. Für die Sighting SROs werden drei Aspekte definiert:

- Was wurde gesichtet (Repräsentiert durch ein Malware oder Indicator Objekt)?
- Wer oder wo wurde es gesichtet (Repräsentiert durch ein Identify Objekt)?
- Was wurde tatsächlich gesehen (repräsentiert durch ein Observed Data Objekt)?

Sighting-Beziehungen unterscheiden sich von Observed Data Objekten darin, dass sie eine Behauptung darstellen, während Observed Data eine konkrete Information ist. Daraus lässt sich die aufgestellte Behauptung nachweisen, indem eine Beziehung von einem Observed Data SDO mittels einer Sighthing SRO zu einem anderen SDO hergestellt werden kann. Ein Beispiel ist in Abbildung 2.2 zu finden. [28]

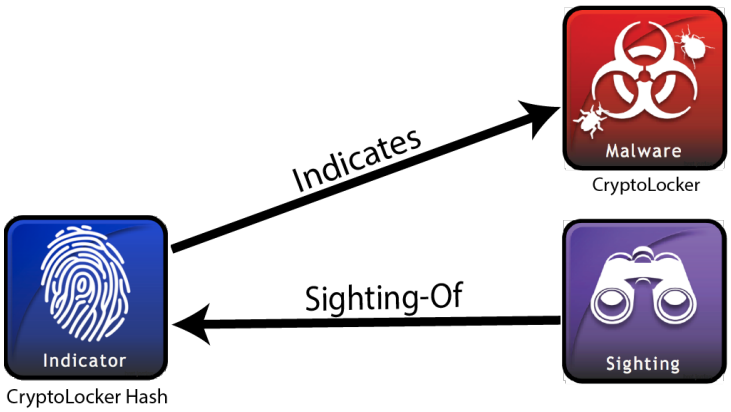

Abbildung 2.2
Beschreibung eines Indicator durch ein Sighting STIX Relationship Object [70]

Incident Object Description Exchange Format (IODEF)

IODEF ist ein weiterer Standard zum Austausch von CTI. Er ist im Requests for Comments (RFC) 7970 definiert und wird von vielen Plattformen unterstützt [80, 7]. Das Datenmodell zur Repräsentation der Informationen in IODEF wird in XML beschrieben. Der Standard wird hauptsächlich von Computer Security Incident Response Teams (CSIRTs) eingesetzt und soll u.a. Bedrohungen durch die Beschreibung von Indicators

charakterisieren oder Sicherheitsvorfälle dokumentieren. Dadurch sollen CSIRTs Bedrohungen besser verstehen sowie auf konkrete Vorfälle effektiver reagieren können. Infolgedessen können die Informationen zwischen verschiedenen Organisationen ausgetauscht und ein proaktives Abschwächen der Vorfälle erreicht werden. Das Datenmodell unterteilt sich in eine Vielzahl von Klassen, welche seit der vorherigen Version von IODEF gewachsen ist. Der Standard aus dem RFC 5070 war hauptsächlich zur Darstellung von Vorfallsberichten erstellt worden. Die neue Version wurde grundsätzlich zur Darstellung von Indicators ergänzt. Aber auch weitere Herausforderungen mit denen die CSIRTs in den letzten Jahren konfrontiert waren, werden durch diverse Ergänzungen und Änderungen an dem Standard gekennzeichnet. In IODEF sind verschiedene einfache und komplexere Datentypen sowie Klassen definiert. Nachfolgend werden einige Klassen und ihre Beziehungen zu Datentypen sowie untereinander kurz beschrieben. Wohingegen Beziehungen als Elemente im XML-Kontext zu betrachten sind. [7]

Document Class ist die Wurzelklasse jedes Datenmodells, sodass jedes IODEF Dokument ein Objekt dieser Klasse ist. Diese Klasse hat eine oder mehrere Beziehungen zu Incident Classes sowie keine oder mehrere Beziehungen zu AdditionalData Classes. Eine AdditionalData Class wird zur Informationsanreicherung des Datenmodells genutzt. Dies wird durch den Datentyp Extension erreicht, welcher Informationen enthält die nicht anderweitig durch eine IODEF Klasse abgedeckt sind. Zudem besitzt die Document Class u.a. die Attribute *version* - nach dem RFC 7970 immer *2.00* - und *xml:lang* - eine Auflistung von Sprachen in dem das Dokument verfasst ist. [7]

Eine *Incident Class* beschreibt Informationen über einen konkreten Sicherheitsvorfall. Sie weist eine Beziehung zu genau einer IncidentID Class auf, wodurch der Vorfall eindeutig identifizierbar ist. Zudem ist eine Beziehung zu genau einer GenerationTime notwendig, die die Erstellungszeit des Vorfalls definiert. Mit der Beziehung zu einer oder mehreren Contact Classes werden die Kontaktdaten involvierter Parteien beschrieben. Ferner kann eine DetectTime, StartTime oder EndTime eines Vorfalls in dem Dokument angegeben werden. Durch eine Verbindung zu einer oder mehrerer Assessment Classes können Bewertungen des Vorfalls abgegeben werden. Zudem kann eine Beziehung zu einer IndicatorData Class vorhanden sein. Diese beschreibt durch eine Verbindung zu einer oder mehrerer Indicator Classes weitere Hinweise auf den Vorfall. Mittels der verpflichtenden Angabe des Attributes *purpose* wird die Begründung zur Dokumentation dieser Informationen gekennzeichnet. Mögliche Werte sind u.a. *traceback*, *mitigation* oder *reporting*, um den Vorfall zurück zu verfolgen, ihn abzuschwächen oder zu berichten. Daneben kann der Status des Vorfalls durch das Attribut *status* u.a. mit *new*, *forwarded* oder *resolved*

angegeben werden. [7]

Zur Beschreibung eines Threat Actors gibt es die gleichnamige *ThreatActor Class*. Durch keine oder mehrere Beziehungen zu ThreatActorIDs, URLs, Descriptions oder AdditionalData wird ein Threat Actor genauer spezifiziert. Allerdings muss eine Instanz dieser Klasse mindestens eine der aufgeführten Klassen enthalten. [7]

Campaign Class definiert eine Angriffskampagne die durch einen Threat Actor durchgeführt wird. Die Klasse besitzt die gleichen Beziehungen wie eine ThreatActor Class und eine Instanz muss ebenso mindestens eine der aufgeführten Klassen enthalten. [7]

Damit die Auswirkungen eines Vorfalls auf ein Opfer beschrieben werden kann, besitzt IODEF die *Assessment Class*. Innerhalb dieser Klasse werden fünf weitere Klassen spezifiziert: SystemImpact, BusinessImpact, TimeImpact, MonetaryImpact und IntendedImpact. Diese können gar nicht oder mehrfach innerhalb der Assessment Class vorkommen. Eine dieser Klassen muss aber mindestens vorhanden sein. Die SystemImpact Class beschreibt die technischen Auswirkungen des Vorfalls auf das System oder Netzwerk. Durch das Attribut *type* kann die Auswirkung z.b. mit dem Wert *takeover-system* oder *traffic-redirection* präzisiert werden. Die Auswirkungen auf die Tätigkeiten einer Organisation werden durch die BusinessImpact Class festgelegt. Durch eine Beschreibung und dem Attribut *type*, welcher u.a. die Werte *loss-of-service* oder *theft-financial* annehmen kann, wird die Auswirkung konkret benannt. IntendedImpact Class ist exakt wie die BusinessImpact Class aufgebaut, beschreibt aber die Absichten eines Threat Actors und weniger die bereits eingetretene Auswirkung. Zur Dokumentation der Ausfallzeit oder der Zeit zur Stabilisierung eines Systems, kann die TimeImpact Class genutzt werden. Zur genaueren Beschreibung einer Auswirkung wird eine konkrete Zeit angegeben und u.a. das Attribut *severity* zur Angabe der Schwere einer Auswirkung. Der finanzielle Schaden wird in der MonetaryImpact Class festgehalten. Ebenso zur detaillierteren Beschreibung gibt es einen konkreten Wert sowie die optionale Angabe der Schwere. [7]

Zur Dokumentation von Indicators besitzt IODEF die *Indicator Class*. Darin werden beobachtete Artefakte zur Unterstützung in der Forensik oder der proaktiven Erkennung von bösartigem Verhalten festgehalten. Die Klasse besitzt keine oder eine Beziehung zu der Observable Class. Innerhalb dieser wird bösartiges Verhalten dokumentiert. Diese Klasse kann u.a. eine Beziehung zu den bereits beschriebenen Incident oder Assessment Classes besitzen. Weitere Klassen sind z.b. DomainData oder FileData. Zudem besitzt eine Indicator Class genau eine IndicatorID zur genauen Identifizierung. Ein Attribut zur StartTime und EndTime sowie mehrere Beschreibungen sind ebenso möglich. [7]

IODEF spezifiziert noch einige weitere Klassen, sodass ein Vorfall beliebig komplex beschrieben werden kann. Diverse Beispiele für IODEF-Dokumente sind in dem RFC aufgeführt. Des Weiteren wird dort explizit aufgeführt, dass das Datenmodell keine Sicherheits- und Datenschutzvorgaben macht. Daher ist jede involvierte Partei bei dem Austausch dieser Informationen angehalten darauf Acht zu geben, welche Daten untereinander ausgetauscht werden. [7]

MISP Core Format

Im Folgenden wird das Kernformat für die Repräsentation von Bedrohungsinformationen in MISP vorgestellt. Eine genauere Vorstellung von MISP wird in Kapitel 2.4 vorgenommen.

Das Kernformat wird durch das JSON-Format abgebildet. Großgeschriebene Schlüssel in einem JSON-Dokument, wie z.B. *Event* oder *Attribute*, stellen ein Datenmodell in Form eines JSON-Objektes dar. Kleingeschriebene Schlüssel sind Attribute des jeweiligen Datenmodells. Die Datenmodelle sowie ihre jeweiligen Attribute die nun vorgestellt werden stellen einen kleinen Ausschnitt dar. Für eine umfassendere Beschreibung sei hier auf die Dokumentation verwiesen. [11]

Ein *Event* Datenmodell stellt eine Struktur zur Darstellung von Indicators mithilfe von Attributen und weiteren Metadaten dar. Ein Event kann aus verschiedenen Quellen gebildet worden sein, wie bspw. aus einem Vorfall oder einer Analyse eines Sicherheitsberichts. Daher wird die Bedeutung eines Events durch seine Attribute definiert. Das Attribut *uuid* ist eine nach dem RFC 4122 standardisierte eindeutige Kennung für das Event [31]. Wohingegen das Attribut *id* eine menschenlesbare Identifikationsnummer für die eigene Instanz ist. *info* stellt eine Zusammenfassung des Events dar und muss zudem für jedes Event vorhanden sein. Gleiches gilt für das Attribut *date*, welches das Datum des Auftretens des Events angibt. *threat_level_id* spiegelt das Level der Bedrohung wider und kann die Werte *undefined*, *low*, *medium* und *high* annehmen. Zur Kontrolle über die Verteilung eines Events gibt es das Attribut *distribution*. Somit kann das Event über die eigene Community oder Organisation geteilt werden. [11]

Attributes sind gesonderte JSON-Objekte, welche ein Event genauer beschreiben. Die wichtigsten Informationen zu einem Event sind durch drei Schlüssel-Wert-Paare abgebildet. *category*, *type* und *value* sind die jeweiligen Schlüssel. Wohingegen *category* und

type die Bedeutung sowie den Kontext von *value* vorgeben und nur in bestimmten Kombinationen auftreten dürfen. Die Ausschnitte 2.1 und 2.2 zeigen je ein Beispiel für ein *Attribute* mit einer validen Kombination von *category* und *type*. Weitere korrekte Kombinationen können der Dokumentation entnommen werden. Wie ein Event, besitzt ein *Attribute*-Objekt ebenfalls eine eindeutige *uuid* und *id* sowie ein Attribut *distribution* zur Regelung der Verteilung. *event_ id* stellt die Relation zu dem zugehörigen Event her. Mithilfe des Attributs *data*, kann ein base64-kodiertes Malware-Sample dem Event zugeteilt werden. Durch RelatedAttribute kann eine Liste korrelierter *Attributes* mit diesem Attribut angegeben werden. [11]

```
{
        "category": "Network activity",
        "type": "ip-dst",
        "value": "127.0.0.1"
}
```

Codeausschnitt 2.1
Erster Ausschnitt für ein valides MISP-Attribute

```
{
        "category": "Person",
        "type": "first-name",
        "value": "Fabian"
}
```

Codeausschnitt 2.2
Zweiter Ausschnitt für ein valides MISP-Attribute

SchadowAttributes können von 3rd Parties erstellt werden, um einem Event weitere Informationen zu geben oder bestehende zu modifizieren. Nach Zustimmung des Eventerstellers werden diese *Attributes* in gewöhnliche *Attributes* umgewandelt. Wie *Attributes* verfügen *SchadowAttributes* über nahezu die gleichen Attribute: *uuid*, *id*, *category*, *type*, etc.. Zusätzlich besitzen diese *Attributes* eine *org_ id* zur Identifikation der Organisation, welche das ShadowAttribute erzeugt hat. [11]

Eine komplexere Beschreibung von Informationen zu einem Event kann mit einem *Object* erreicht werden. Innerhalb eines Event werden mehrere *Attributes* zu einer Liste zusammengefasst. Ein Object wird mithilfe eines Templates erzeugt, durch einen Namen benannt und durch eine frei wählbare Meta-Kategorie kategorisiert. Diese Kategorie wird durch das Attribut *meta-category* dargestellt. Somit können z.B. zu einer Datei

verschiedene Informationen eines Events, wie Dateiname oder Hashwert übersichtlich gebündelt werden. *template_uuid* und *template_version* spezifizieren das zur Erstellung des Objects genutzte Template. [11]

Object References bieten die Möglichkeit einer logischen Beziehung zwischen einem Object und einem anderen Object oder *Attribute*. Die Benennung der Beziehung ist durch eine Liste von vordefinierten Bezeichnungen gegeben und wird durch das Attribut *relationship_type* bestimmt [42]. Beispielsweise wird durch *duplicate-of* eine semantische Doppelung zweier Objects bzw. einem Object und einem *Attribute* angegeben. *object_id* und *object_uuid* spezifizieren ID bzw. die UUID des ursprünglichen Objects. Während *referenced_id* sowie *referenced_uuid* jene Referenz auf das zu referenzierende Object bzw. *Attribute* sind. Das Attribut *reference_type* gibt an, ob die Beziehung zu einem *Attribute* oder Object besteht und wird mit *0* bzw. *1* angegeben. [11]

Zur Klassifizierung von Events oder *Attributes* können *Tags* eingesetzt werden. Sie werden durch einen Namen benannt, welcher frei wählbar ist oder aus der Liste von bereitgestellten Taxonomien ausgesucht werden kann [43]. Werden die Events über die Grenzen der eigenen Organisation geteilt, wird zur Konsitenzwahrung empfohlen die Tags aus den Taxononmien zu wählen. [11]

Durch *Sightings* können *Attributes* als beobachtet markiert werden. Zudem kann die Organisation angegeben werden, welche das *Attribute* beobachtet hat. Das Attribut *type* unterscheidet drei Fälle in denen beschrieben werden kann, ob das *Attribute* gesichtet wurde, ob es gesichtet wurde und als *false positive* zu werten ist und ob das *Attribute* mit der Sichtung verfällt. Zusätzlich muss *uuid* und *date_sighting* angegeben werden. Die Quelle mit der das *Attribute* gesichtet wurde, z.B. durch einem Security Information and Event Management (SIEM), sowie *event_id* und *attribute_id* sind optional. [11]

Weitaus größere Objects, sogenannte Cluster, können mit einer MISP *Galaxy* beschrieben und den Events oder *Attributes* beigefügt werden. In der Galaxy werden neben den Clustern u.a. eine kurze Beschreibung und die Quelle der Cluster bereitgestellt. Es gibt bereits eine Liste von Clustern die in MISP genutzt werden können [36]. Eigene Cluster können zudem erstellt werden. Der Android-Cluster bspw. stellt Informationen zu bekannter Malware für Android bereit [37]. [11]

2.3.2 Protokolle

Datenaustauschprotokolle für CTI können nach Kampanakis [29] in synchrone und asynchrone Modelle eingeteilt werden. Ein synchrones Modell sieht vor, dass ein Client die Informationen von einem Server abfragt (*pull*-Methode). Wohingegen in einem asynchronen Modell der Server die Informationen an einen Client übermittelt, sofern dieser sein Interesse bekundet hat (*push*-Methode). Dieses Modell wird zudem als *Publish/ Subscribe*-Muster bezeichnet.

In diesem Unterkapitel werden die Protokolle TAXII und RID genauer erläutert. TAXII besitzt einen synchronen sowie asynchronen Mechanismus zum Informationsaustausch, wohingegen RID einen synchronen Mechanismus umsetzt [29]. Es wurde bereits erwähnt, dass MISP ein eigenes REST-API für den Informationsaustausch besitzt. REST wird hier allerdings nicht vorgestellt, da es kein spezifischer Standard im CTI-Kontext ist.

Trusted Automated eXchange of Indicator Information (TAXII)

TAXII ist ein Protokoll für den Austausch von CTI über HTTPS. Durch die angebotenen Dienste *Collections* und *Channels*, werden verschiedene Austauschmodelle unterstützt. TAXII wurde wesentlich für den Austausch von Informationen die durch STIX repräsentiert sind konzipiert. Allerdings ist es nicht darauf beschränkt. Im Folgenden wird der Aufbau des Protokolls kurz beschrieben. [97]

Ein TAXII-Server kann mehrere *API Roots* besitzen. Mit ihnen können logisch zusammengehörige Channels und Collections zusammengefasst werden. Dadurch werden Informationen konsequent separiert und der Zugriff darauf klar definiert. Der Zugriff auf diese Dienste erfolgt über sogenannte *Endpoints*. Ein Endpoint wird über eine spezifische URL und einer HTTP-Methode definiert. So wird bspw. Über eine GET-Anfrage auf die URL *<api-root>/collections/id/* eine über die ID definierte Collection abgerufen. Für jeden Endpoint ist dokumentiert, welchen Anforderungen sie entsprechen müssen. Erfolgt bspw. eine erfolgreiche Antwort der obigen Anfrage, muss der Content-Type der HTTP-Antwort *application/vnd.oasis.taxii+json;* und der Body vom Datentyp *collection* sein. Das Filtern oder Sortieren jedes Endpoints ist zusätzlich genau spezifiziert. [97]

Collections dienen zur Speicherung von CTI. Es können mehrere Collections innerhalb eines API Roots auf einem

TAXII-Server abgelegt werden. Sie werden mithilfe eines TAXII-Clients in Form eines Request-Response-Modells angelegt oder abgerufen. Abbildung 2.3 verdeutlicht dies. [97]

Mithilfe von Channels werden CTI durch das Publish/Subscribe-Modell ausgetauscht, wie Abbildung 2.4 verdeutlicht. Dies ermöglicht den TAXII-Clients das Veröffentlichen und Konsumieren von Bedrohungsinformationen. Ebenfalls kann ein TAXII-Server mehrere Channels in einem API Root besitzen. Eine genauere Spezifikation von Channels ist in der aktuellen TAXII Version noch nicht enthalten. [97]

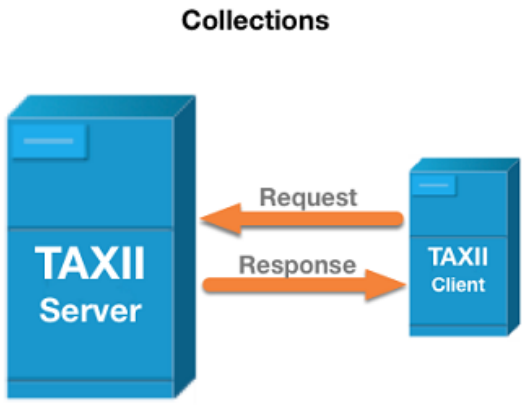

Abbildung 2.3
Collections-Austauschmodell in TAXII [71]

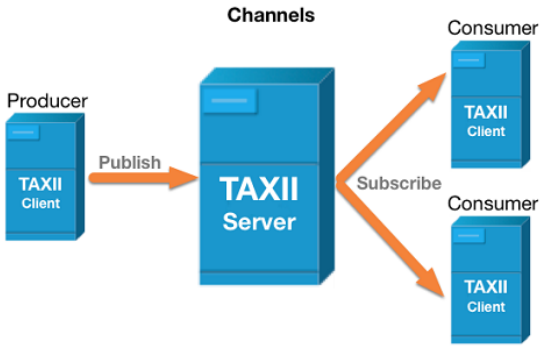

Abbildung 2.4
Channels-Austauschmodell in TAXII [71]

Real-time Inter-network Defense (RID)

Das nach dem RFC 6545 definierte RID stellt ein Schema zur Verfügung, womit kommunikationsspezifische Metadaten für den Austausch von CTI, die bspw. im IODEF-Format repräsentiert sind, definiert werden [52]. Der Transport von RID-Nachrichten über HTTP/TLS wird im RFC 6546 als Protokoll der Anwendungsschicht spezifiziert [93]. Das Schema stellt somit ein Rahmen zur konkreten Beschreibung von Informationen dar, welches zudem durch ein speziell definiertes Transportprotokoll zwischen verschiedene Parteien ausgetauscht werden kann. Daher werden diese beiden Komponenten zusammen in diesem Abschnitt beleuchtet.

RID-Schema

Die Aufgabe von RID ist die Weitergabe konkreter technischer Sicherheitsvorfälle, um weiteren Angriffen vorzubeugen bzw. anderen zu ermöglichen vorbeugende Maßnahmen zu treffen. Die Daten werden zwischen CSIRTs, Service Providern sowie Regierungen oder Unternehmen untereinander ausgetauscht. Weiterhin integriert es Mechanismen zum Erkennen von Vorfällen zur Identifikation der Quelle sowie zum Abschwächen möglicher Auswirkungen. Folglich soll mit RID eine ganzheitliche Lösung zur Behandlung von Vorfällen zur Verfügung gestellt werden. [52]

Neben IODEF als Datenmodell können Vorfälle zudem in selbst definierten XML-Modellen beschrieben werden. Diese müssen vor der Verwendung zunächst bei der Internet Assigned Numbers Authority (IANA) registriert werden. Weiterhin werden Sicherheits-, Policy-, und Datenschutzaspekte im Bezug auf den Austausch von sensiblen Informationen berücksichtigt. RID enthält zudem Mechanismen die die Authentizität, Vertraulichkeit sowie Integrität der auszutauschenden Sicherheitsvorfälle berücksichtigen. [52]

RID-Nachrichten beschreiben einen konkreten Vorfall und werden zwischen involvierten Parteien ausgetauscht. Durch diesen Mechanismus können direkte Maßnahmen zum Unterbinden des Vorfalls ergriffen werden. Ist beispielsweise die Quelle eines Denial of Service (DoS)-Angriffs bekannt, kann dieser geblockt oder die Bandbreite eingeschränkt werden. Die Erkennung eines Vorfalls ist laut den Autoren des RFCs 6545 allerdings mit einigen Schwierigkeiten verbunden. Unter anderem kann ein Angriff von verschiedenen Quellen ausgeführt worden sein oder der auftretende Datenverkehr kann nicht eindeutig zwischen validen oder bösartigem Verkehr unterschieden werden. Mit IODEF und RID

können die Vorfälle detailliert dokumentiert werden. RID-Nachrichten werden in fünf Typen unterteilt und können Beziehungen zu den drei in RID definierten Klassen aufweisen - jenes Klassenprinzip welches bereits in IODEF definiert ist. Die folgenden Abschnitte erörtern die Typen und Klassen etwas genauer. [52]

Der Nachrichtentyp *Request* dient der Anfrage zur Unterstützung von möglichen Sicherheitsvorfällen und wird in zwei spezifische Typen unterteilt: *InvestigationRequest* und *TraceRequest*. Nachrichten vom Typ InvestigationRequest werden an den nächstgelegenen Service Provider versandt, um diesen über einen möglichen Vorfall in Kenntnis zu setzen. Dadurch können Nachforschungen zu der im Request enthaltenen Quelle durchgeführt werden. TraceRequest-Nachrichten dienen der schrittweisen Verfolgung durch das Netzwerk zurück zur Quelle des bösartigen Datenverkehrs. Eine Request-Nachricht muss in ihrer Struktur die RIDPolicy Class mit diversen Attributen und Beziehungen zu Klassen aufweisen, wie z.B. dem Attribut MsgType oder der IncidentID Class. Zusätzlich muss das IODEF-Dokument gewisse Informationen bereitstellen, wie bspw. einem Incident Identifier bestehend aus der Incident Class und IncidentID Class oder einem Confidence Rating betsehend aus einer Impact und Confidence Class. [52]

Acknowledgement-Nachrichten dienen der Bestätigung des Erhaltes von Nachrichten die vom Typ Request, Report oder Query sind. Darin wird der Status einer vorhergehenden Nachricht festgehalten oder ein Nachweis aufgeführt warum die Anfrage nicht durchgeführt werden konnte. Dieser Nachrichtentyp muss ebenfalls eine Beziehung zur RIDPolicy Class aufweisen sowie eine zu der RequestStatus Class. [52]

Mithilfe von *Result*-Nachrichten wird das Ergebnis in Form eines Berichts zu einer abgegebenen Request-Nachricht übermittelt. Darin wird angegeben, ob eine angefragte Quelle ermittelt werden konnte. Zusätzlich sollte mitgeteilt werden, welche Maßnahmen zur Ermittlung vorgenommen wurden. Die Struktur einer Result-Nachricht weist eine RIDPolicy Class sowie eine IncidentSource Class und ein IODEF-Dokument mit diversen Informationen auf. [52]

Report-Nachrichten dienen der Meldung eines Sicherheitsvorfalles, sowohl, ohne dass eine Partei eine Anfrage gestellt haben muss, als auch als Antwort auf eine Request- oder Query-Nachricht. Dadurch können Informationen zu Vorfällen oder Statistiken unaufgefordert miteinander geteilt werden. Es ist eine Beziehung zu einer RIDPolicy Class anzugeben. Informationen im IODEF-Format werden empfohlen, sind aber nicht zwingend erforderlich. [52]

Durch *Query*-Nachrichten können Informationen zu einem konkreten Vorfall angefragt werden. Die Informationen werden üblicherweise durch eine Report-Nachricht repräsentiert. Ist eine IncidentID Class - innerhalb der notwendigen RIDPolicy Class - vorhanden, kann der Versandt der Antwort automatisiert werden. Sind hingegen detaillierte Informationen vorhanden - dargestellt durch ein IODEF-Dokument -, muss ggf. manuell eine Antwort erstellt und versandt werden. Sobald beispielsweise nach einem speziellen Vorfallstypen innerhalb einer Query-Nachricht abgefragt wird, können mehrere Vorfälle in der Report-Nachricht aggregiert werden. [52]

Die *RequestStatus Class* wird nur in Acknowledgement-Nachrichten verwendet und spiegelt den Status einer Request-Nachricht wider. Eine *IncidentSource Class* wird nur in Result-Nachrichten integriert und stellt Informationen über die ermittelte Quelle des Vorfalls bereit. Mithilfe der *RIDPolicy Class* werden in allen Nachrichtentypen die vereinbarten Richtlinien zwischen den involvierten Parteien festgehalten. Außerdem kann maximal ein IODEF-Dokument durch das Element ReportSchema in diese Klasse integriert werden. Der Grund für ein separates RID-Schema ist die Flexibilität beim Versand von RID-Nachrichten die dadurch geboten wird. Zusätzlich werden die Sicherheitsaspekte die RID bietet, sowohl auf das IODEF-Dokument, als auch auf die RID-Nachrichten angewendet. [52]

RID-Austauschprotokoll

Wie in einem der vorherigen Abschnitte bereits geschildert, wird im RFC 6546 das Protokoll *Transport of Real-time Inter-network Defense Messages over HTTP/TLS* für den Austausch von RID-Nachrichten über HTTP/TLS spezifiziert. Demnach fungiert jeder RID-Server sowohl als Server als auch als Client und muss infolgedessen Anfragen über HTTP/TLS versenden und akzeptieren können. Zur Vermeidung von Irritationen wurde entsprechend der TCP-Port 4590 für den Empfang und Versand von Anfragen reserviert. Es sollten daher alle RID-Server auf diesem Port ihren Dienst zur Verfügung stellen. Ferner muss ein RID-Server alle Funktionen von HTTP/1.1 nach RFC 2616 umsetzen. Alle RID-Nachrichten sind zudem über eine POST-Anfrage zu übermitteln. GET- sowie HEAD-Anfragen werden mit dem HTTP-Statuscode *204 No Content* beantwortet. Aus Sicherheitsgründen sollten keine Antworten mit einem 3xx-Statuscode zur Umleitung von Anfragen erzeugt und auf keine Antworten aus dieser Kategorie reagiert werden. Wird eine unzulässige RID-Nachricht in einer HTTP-Anfrage empfangen, muss eine entsprechende Antwort mit einem 4xx-Statuscode zurückgeschickt werden. Kommt es zu einem

Serverfehler muss ein angemessener 5xx-Statuscode in der Antwort enthalten sein. Des Weiteren enthält die Spezifikation einen Callback-Mechanismus, sofern zusammengehörige RID-Nachrichten eine zu hohe zeitliche Differenz aufweisen. Generell gilt, dass jede initiale RID-Nachricht eine Antwort erwartet, sei es die direkte Antwort oder ein Callback darauf. Auf welche RID-Nachricht welche andere Nachricht zu folgen hat, ist im RFC tabellarisch dargestellt. Sofern RID-Nachrichten über HTTPS versandt werden, sind TLS 1.1 (MUST) und 1.2 (SHOULD) im RID-Server zu implementieren. [93]

2.4 Austausch von CTI über Plattformen

Skopik u. a. [86] gibt eine ausführliche Übersicht zu nationalen und internationalen Organisationen sowie Verbünden an, die den Austausch von CTI befördern. Regionale CERTs sind hierbei unerlässliche Unternehmen zum Sammeln von Bedrohungsinformationen und einer frühzeitigen Warnung gegen Bedrohungen. Sie stehen zugleich international in engem Kontakt miteinander. Forum for Incident Response and Security Teams (FIRST) ist eine weitere Organisation die sich für eine bessere Kommunikation und Koordination zwischen IRTs einsetzt. Sie ist mit mehr als 400 Teams in über 60 Ländern aktiv [21]. Für eine bessere Reaktion auf Vorfälle tauschen sich die Teams u.a. über den Einsatz von Werkzeugen, bewährten Methoden oder praktischen Erfahrungen aus. [86]

Ein zentraler Bestandteil für den Austausch von CTI sind sogenannte Threat Intelligence Platforms (TIPs). Dadurch ist es für Unternehmen möglich, ihr Wissen über potentielle Bedrohungen zu erweitern und sich effektiver gegen Angriffe zu schützen [15]. Für eine ausführliche Übersicht vorhandener und etablierter Plattformen sei auf die European Union Agency For Network and Information Security (ENISA) [15] und Reiber [79] verwiesen.

Der folgende Abschnitt zeigt einen detaillierteren Überblick über MISP als eine von mehreren TIPs auf. MISP hat sich, wie Shackleford [83] aufzeigt, seit der Veröffentlichung immer weiter verbreitet. Es wird mittlerweile in mehr als 25% der befragten Regierungen und Unternehmen, bspw. aus dem Banken- oder Telekommunikationsbereich, eingesetzt.

Im daran anschließenden Abschnitt wird eine kurze Einführung zur Exploit Database (ExploitDB) von Offensive Security [63] gegeben. Dabei handelt es sich um eine öffentlich

zugängliche Datenbank für den Austausch von Exploits, welche für die in dieser Arbeit vorgestellten Analysesoftware genutzt wird.

Malware Information Sharing Platform

MISP ist eine im Jahr 2011 entwickelte quelloffene Plattform für den Austausch von Indicator of Compromise und Bedrohungsinformationen. Diese Informationen können bspw. aus dem Finanzsektor oder dem Bereich IT-Sicherheit kommen. Dadurch wird ein Austausch über verschiedene Bereiche hinaus gewährleistet, sodass ein besseres Verständnis zum Thema Sicherheit gefördert werden kann. In einem der vorherigen Unterkapitel wurde bereits das Kernformat zur Repräsentation von Bedrohungsinformationen in MISP vorgestellt. In den folgenden Absätzen wird der Austausch dieser Informationen genauer erörtert. [96]

MISP begegnet mit dem gewählten Austauschmodell der Ambivalenz zwischen der Geheimhaltung von Informationen und dem Gewinnen neuer Erkenntnisse durch ausgetauschte Informationen. Zu jedem Event kann eine Verteilungsregel gewählt werden, wodurch sichergestellt wird mit welcher Zielgruppe das Event ausgetauscht werden soll. Mit der Option *organization only*, ist das Event nur für Mitglieder der eigenen Organisation mit Zugriff auf dem eigenen Server einsehbar. *community only* umfasst alle Organisationen die Teil einer MISP-Community sind. *connected communities* definiert die Einsicht von Events aller Organisationen die entweder einer MISP-Community angehören oder Teil einer direkt verbundenen Community sind. Mit der Angabe von *all* wird das Event mit allen MISP-Communities ausgetauscht. Eine weitere Besonderheit für den Austausch von Events stellen die sogenannten *Proposals* dar. Ein Event kann ausschließlich von der Organisation modifiziert werden, die es erstellt hat. Andere Organisationen können sodann ein Proposal zu einem Event einer anderen Organisation erzeugen, welches darauf hinweist, dass das Event z.B. falsche Informationen enthält oder ein *false positive* ist. [96]

Das Filtern von Events in MISP ist bei einer hohen Anzahl an Events unabdingbar und unterstützt dabei die Klassifizierung dieser. Zur Klassifizierung wurden daher *Tags* eingeführt, welche von Organisationen erzeugt und untereinander ausgetauscht werden können. Allerdings wurden immer wieder viele neue Tags erzeugt und ausgetauscht, sodass die Filterung von Events bei der Vielzahl von Tags komplexer wurde. Daher wurden zur Klassifizierung von Events Taxonomien entwickelt. Eine Taxonomie wird durch die

sogenannte *triple-tag*-Syntax definiert. Sie besteht aus dem *namespace*, *predicate* und dem *value*. Die *Malware Classification* Taxonomy wird beispielsweise zur Klassifizierung von Malware Samples eingesetzt und kann wie folgt aussehen: *malware-classification: malware-category = ‚Virus‘*. *malware-classification* stellt den *namespace* dar, *malware-category* das *predicate* und *Virus* den *value* [38]. Ein Vorteil der Taxonomien ist die leserlichere Form der Klassen. Neben einer Vielzahl von vordefinierten Taxonomien aus den unterschiedlichen Bereichen können Organisationen überdies eigene definieren [43]. [96]

Für den Austausch von Events zwischen verschiedenen miteinander verbundenen MISP-Instanzen, besitzt MISP im wesentlichen drei unterschiedliche Synchronisationsalgorithmen. Sie beruhen auf dem *trial-and-error*-Ansatz, welcher die Ziele Effizienz, Skalierbarkeit und Fehlerfreiheit verfolgt. Mithilfe des *pull*-Algorithmus können MISP-Instanzen neue oder aktualisierte Events, in Abhängigkeit der jeweils definierten Verteilungsregel, von anderen Instanzen heruntergeladen werden. Eine Alternative zur *pull*-Methode stellt das sogenannte *cherry picking* dar. Dadurch können gezielt einzelne Events einer anderen verbundenen Instanz in einer speziellen Oberfläche ausgewählt und heruntergeladen werden. Eine spezielle Synchronisationsfunktion erlaubt es beim *cherry picking*, diese bezogenen Events aktuell zu halten, ohne dass neue Events der Instanz heruntergeladen werden müssen. Der *push*-Algorithmus - als dritte Methode zur Synchronisation von Events - erlaubt es zum einen, dass ausgewählte Events einer MISP-Instanz an eine andere Instanz geschickt werden. Zum anderen kann ein einzelnes Event veröffentlicht werden, sodass der *push*-Algorithmus dieses an alle verbundenen Instanzen verschickt. Neben diesen drei Methoden gibt es zudem das Feed-System. Dadurch können bspw. *air-gapped*-Systeme ebenfalls Events erhalten. Der Austausch aller Events erfolgt immer im JSON-Format. [96]

MISP Objects dienen zur komplexeren Beschreibung von Events. Vordefinierte Objects werden durch Templates beschrieben. Sie kombinieren diverse Attribute miteinander und bündeln die Informationen zu einem Event. Die bereits vorhandenen Objects stammen aus konkreten Anwendungsfällen. Beispielsweise dient das Object *exploit-poc* zur Beschreibung eines Proof of Concept oder eines Exploits zu einer Schwachstelle. Dieser besitzt z.B. die Attribute *description* zur Beschreibung und *author* zur Angabe des Verfassers eines Exploits. [40, 41]

Des Weiteren gibt es sogenannte *MISP Modules*. Damit die Kernfunktionalität von MISP unabhängig bleibt, erweitern diese MISP um weitere Funktionen. Dadurch können zum

einen Events mit weiteren Informationen angereichert oder in der MISP Oberfläche erweitert werden. Zum anderen ist der Import und Export von Daten mithilfe der Modules gelöst. Zur Anreicherung einer vorhandenen CVE in einem Event, ergänzt z.B. das CVE-Module das jeweilige Event mit ausführlichen Informationen darüber. [39, 33, 34]

Exploit Database (ExploitDB)

In der ExploitDB werden Exploits sowie Proof of Concepts (PoCs) zur Ausnutzung von Schwachstellen öffentlich zugänglich gemacht. Sie können über eine E-Mail eingereicht werden oder stammen aus anderen öffentlichen Quellen. Mithilfe dieser Datenbank werden die Exploits und PoCs strukturiert abgespeichert. Durch zusätzliche Suchkriterien kann sich ein zügiger und einfacher Überblick über die Daten verschafft werden. Die auf der Webseite „Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers“ [63] zu findenden Exploits bzw. PoCs werden zudem in vier unterschiedliche Typen eingeteilt: *dos*, *remote*, *local* und *webapps*. Eine genauere Spezifizierung dieser Typen gibt es nicht. Ferner stellt das Kriterium *platform*, wie z.B. Windows, Linux, OSX, Android, etc., eine weitere Kategorisierung dar. Jeder Exploit bzw. PoC besitzt zudem eine eindeutige Identifikationsnummer, einen Titel, ein Datum sowie eine Angabe zu einem Autor oder einer Autorin. Innerhalb der Suchkriterien kann außerdem eine CVE-Nummer, ein Port oder ein *Tag* angegeben werden. Der Tag definiert die Art der Schwachstelle, z.B. SQL Injection oder Cross-Site Scripting (XSS). Diverse Statistiken auf der Webseite bieten überdies eine Übersicht zur quantitativen Entwicklung der Exploits bzw. PoCs an. [63]

Darüber hinaus bietet Offensive Security die Exploits bzw. PoCs in einem Repository auf GitHub an. Diese Quelle wird in der hier vorzustellenden Analysesoftware genutzt, da die Webseite von ExploitDB keine Schnittstelle für den Zugriff auf die Daten bereitstellt. Die bereits erwähnten Metadaten *type*, *platform*, usw., sind in einer CSV-Datei in dem Repository zu finden. Weiterhin ist in dem Repository die Kommandozeilenanwendung *SearchSploit* zu finden. Dadurch können die Exploits bzw. PoCs aus dem Repository lokal durchsucht werden. [68]

Wie bereits erwähnt, können Exploits bzw. PoCs durch eine E-Mail eingereicht werden. Diese werden verifiziert, getestet und sodann in die Datenbank eingepflegt. Dies bedingt einen zeitaufwendigen und mühsamen Prozess, sodass Offensive Security diverse Kriterien zum Einreichen veröffentlicht hat. So sollen u.a. keine Exploits oder PoCs eingereicht

werden, die Admin- oder Root-Rechte benötigen. Ein weiteres Kriterium ist die Angabe diverser strukturierter Metadaten als Kommentar innerhalb eines Exploits bzw. PoCs. Diese Metadaten sind für die Analysesoftware interessant, werden im weiteren Verlauf dieser Arbeit nochmals aufgegriffen und sind in dem Codeausschnitt 5.2 auf Seite 71 aufgeführt. [63]

3 Zielgruppen von Cyber Threat Intelligence

In Kapitel 2.2 wurde der CTI-Lifecycle detailliert beleuchtet und kurz einige der möglichen Zielgruppen erwähnt. Im folgenden Abschnitt werden diese und weitere Zielgruppen genauer betrachtet. Bei der Betrachtung sollen folgende Fragen beantwortet werden:

- Welche Ziele verfolgen die Gruppen?
- Wie verwenden sie CTI und wie profitieren sie davon?
- Produzieren sie weitere Cyber Threat Intelligence?

Da nicht alle Zielgruppen eine geeignete deutsche Übersetzung besitzen, sind die jeweiligen englischen Bezeichnungen angegeben. Falls eine Übersetzung vorhanden ist, wird diese in dem jeweiligen Abschnitt erwähnt.

3.1 Security Operations Center

Ein Security Operations Center (SOC) besteht aus diversen Teams mit verschiedenen Zuständigkeiten. Zimmerman [98] führt eine umfangreiche Liste über die Aufgaben eines SOCs auf. Darin sind u.a. das Sammeln, Erstellen, Korrelieren, Analysieren sowie Verteilen von CTI enthalten. Ferner ist es für das Erstellen von Berichten über aufgetretene Vorfälle sowie für Maßnahmen verantwortlich, um ähnliche Vorfälle zukünftig zu verhindern.

Nach Torres [91] kann ein SOC durch vier Schichten bzw. Teams aufgebaut sein. Die erste Schicht umfasst die *Alert Analysts*, welche Daten zu möglichen Vorfällen aus verschiedenen Quellen sammeln und diese an die zweite Schicht weiterleiten. *Incident Responders* führen in der zweiten Schicht genaue Analysen und Korrelationen der erhaltenen Daten

durch. Ferner erkennen sie, ob und in welcher Form ein System von einem Vorfall beeinflusst wurde.[1] *Subject Matter Expert/Hunter* bilden die dritte Schicht und verfügen über ein tieferes Verständnis u.a. von IT-Forensik und Threat Intelligence. Sie sind zudem in der Entwicklung und Verbesserung von Prozessen zur Erkennung von Bedrohungen involviert. Die vierte Schicht besteht aus dem *SOC Manager*, welcher u.a. für die Verwaltung des Personals und eingesetzter Technologien in den anderen Schichten zuständig ist. Eine weitere Aufgabe stellt die Kommunikation zur Unternehmensführung dar.

Dem SOC stehen diverse Herausforderungen gegenüber. Die einzelnen Teams eines SOC sind häufig mit der Menge an möglichen Vorfällen überfordert. Die Einsichtnahme und Auswahl erfordert viel Zeit, sodass einige Vorfälle gar nicht betrachtet werden, obwohl sie ernsthaft analysiert werden müssten. So ergab eine Studie mit 3600 befragten Unternehmen aus 26 Ländern, dass 44% der täglich gesichteten möglichen Vorfälle nicht bearbeitet werden [6]. Der Einsatz von CTI kann dem entgegenwirken. Zum einen können potentielle Vorfälle mit weiteren externen CTI automatisiert angereichert werden. Zum anderen können mithilfe von Threat Intelligence Plattformen irrelevante Vorfälle gefiltert und den Blick auf die relevanten Vorfälle fokussiert werden. [72]

3.2 Incident Response Team

Die steigende Anzahl von Sicherheitsvorfällen und der durch sie verursachte Schaden, macht die Wichtigkeit von Incident Response Teams (IRTs) deutlich. IRTs sollen Angriffe und Vorfälle schnell erkennen und den Schaden möglichst gering halten. Weiterhin sollen die von einem Vorfall ausgenutzten Schwächen eines Systems erkannt und beseitigt sowie der Zustand des Systems vor dem Vorfall wieder hergestellt werden. Zur Realisierung dieser Aufgaben bedarf es eines strukturierten Arbeitsablaufs. Unter anderem haben das National Institute of Standards and Technology (NIST) und die European Union Agency for Cybersecurity (ENISA) ähnliche Incident Response-Lifecycles mit detaillierter Beschreibung der einzelnen Phasen veröffentlicht. Die ENISA zeigt zudem konkrete Beispiele von Organisationen die den Lifecycle in abgewandelter Form praktisch einsetzen [13]. Der folgende Abschnitt geht kurz auf die jeweiligen Phasen aus der Veröffentlichung des NIST ein und ist in Abbildung 3.1 dargestellt. [5]

[1] Eine ausführlichere Definition von Incident Response erfolgt im nächsten Abschnitt.

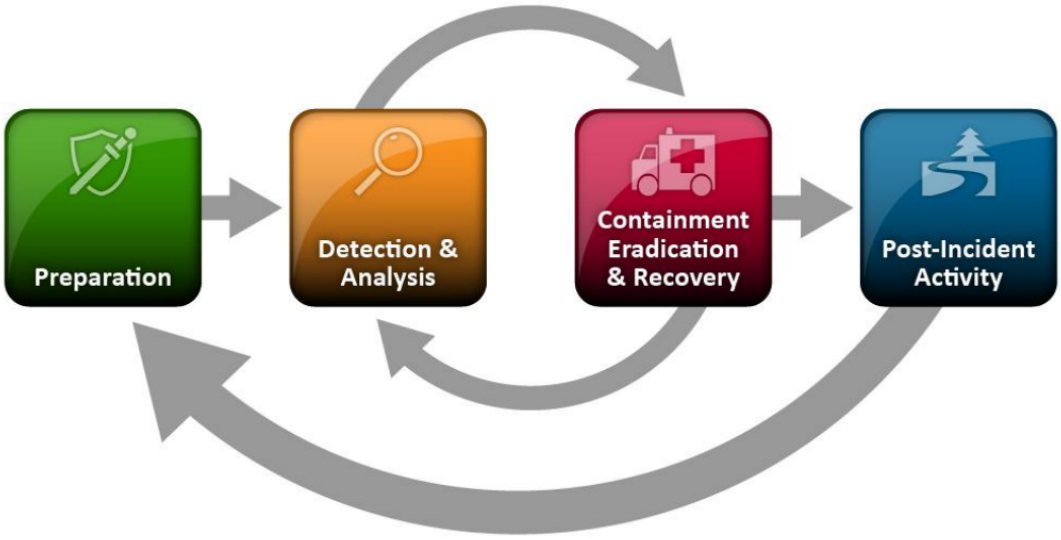

Abbildung 3.1
Incident Response-Lifecycle [5]

Die *Preparation*-Phase dient zum einen zur Vorbereitung zur Vorfallsbearbeitung und zum anderen zur präventiven Verhinderung von Vorfällen. Die Vorbereitung umfasst zusätzlich den Aufbau eines IRT, den Aufbau einer Kommunikationsstruktur innerhalb des Teams sowie der Einrichtung von Soft- und Hardware zur Vorfallsbearbeitung. Das Verhindern von Vorfällen beruht auf einer Bewertung von Risiken, sodass das IRT Empfehlungen zum Schutz des Netzwerkes, der Server und der Endgeräte ausspricht. Es ist zudem für die Sensibilisierung der Mitarbeiter*innen im Hinblick auf Sicherheitsrichtlinien und Verhaltensweisen in der Benutzung der Endgeräte sowie des Netzwerkes zuständig. [5]

Sicherheitsvorfälle können an unterschiedlichen Stellen auftreten. Um diese innerhalb der *Detection & Analysis*-Phase zu erkennen, bedarf es an Wissen über verschiedene Angriffsvektoren. Darunter fallen bspw. Angriffe die durch einen bösartigen E-Mail Anhang oder eines infizierten USB-Sticks ausgelöst werden. Zur Identifizierung möglicher Vorfälle kann verschiedenen Hinweisen nachgegangen werden. Dabei wird vom NIST zwischen *precursors* und *indicators* unterschieden. Hinweise in der *precursor*-Kategorie deuten auf einen möglichen Vorfall in der Zukunft hin, wohingegen Hinweise aus der *indicator*-Kategorie auf einen bereits eingetretenen oder gerade auftretenden Vorfall hindeuten. Die Hinweise können aus verschiedenen Quellen stammen wie z.B. aus Log-Dateien oder öffentlich zugängliche Quellen über Schwachstellen oder Exploits. Während der Vorfallsanalyse

müssen die Hinweise validiert werden, da sie nicht zwingend korrekt sein müssen. Die umfangreiche Menge von Hinweisen und die Notwendigkeit zügig auf konkrete Vorfälle reagieren zu müssen, bedingt eine effektive und effiziente Vorgehensweise beim Analysieren und Eingrenzen relevanter Hinweise. Beispielsweise können die Daten nach bedeutenden Kategorien gefiltert oder Unterstützung von CERTs eingeholt werden. Nachdem ein relevanter Vorfall erkannt wurde, sollte dieser strukturiert und detailliert dokumentiert werden. Zudem sollten die Vorfälle zur weiteren Bearbeitung priorisiert werden, da sie unterschiedlich schwere Auswirkungen haben können. Zuletzt sollten alle Beteiligten über den Vorfall informiert werden. [5]

Die darauf folgende Phase befasst sich zunächst mit dem Finden eines passenden Umgangs mit den Auswirkungen eines Vorfalls. Dies umfasst u.a. das Treffen von wichtigen Entscheidungen und sollte in unterschiedlich vordefinierten Strategien festgehalten sein. Es stellen sich zum Beispiel die Fragen, ob ein System abgeschaltet werden oder ob ein Vorfall weiter beobachtet werden sollte, um weitere Informationen zu erhalten. Daran anschließend müssen die betroffenen Systeme erkannt und mögliche Infektionen entfernt werden. Ferner sollten die ausgenutzten Schwachstellen identifiziert und beseitigt werden, um zukünftige Vorfälle zu unterbinden. Zusätzlich müssen die betroffenen Systeme in einen funktionsfähigen Zustand zurück geführt werden. Gegebenenfalls müssen Patches installiert oder Passwörter geändert werden. [5]

Die *Post-Incident Activity*-Phase dient der regelmäßigen Reflexion der Arbeitsabläufe sowie Erfahrungen mit den aufgetretenen Vorfällen oder neu eingesetzter Werkzeuge. Die dafür vorgesehenen Meetings sollten gut strukturiert sein und diversen Fragestellungen nachgehen. Die daraus gewonnenen Erkenntnisse sollten sodann in die zukünftigen Arbeiten des IRT berücksichtigt werden, womit der Lifecycle in die erste Phase mündet. [5]

Aus dem Lifecycle ist zu erkennen, dass ein IRT zum einen CTI zur Analyse von Vorfällen konsumiert. Zum anderen produziert es aus den Analyseergebnissen neue CTI, um diese weiter verwenden oder mit anderen Organisationen austauschen zu können. Weiterhin ist aus der Verarbeitung von CTI und Vorfällen zu erkennen, dass ein IRT und SOC Ähnlichkeiten zueinander aufweisen. Wie bereits gezeigt ist ein IRT Bestandteil eines SOC. Allerdings weist ein SOC weitere Kompetenzen als ein IRT auf.

Nach Pace u. a. [72] sind IRTs mit diversen Herausforderungen konfrontiert. In den letzten zwei Jahrzehnten ist der Umfang von Sicherheitsvorfällen kontinuierlich gestiegen. Weiterhin werden die Bedrohungen immer komplexer, damit schwieriger zu analysieren und

einzudämmen. Daraus ergibt sich zudem ein Anstieg der aufzubringenden Zeit zur Verarbeitung der Vorfälle. Um diesen Problemen entgegen zu gehen, wird sehr viel Erfahrung benötigt. Allerdings hat sich in der Vergangenheit gezeigt, dass der Fachkräftemangel zugenommen hat und sich dieser in naher Zukunft nicht verringern wird. Pace u. a. [72] bewertet den IR-Lifecycle als sehr reaktiv. Dies bedeutet, dass bereits sehr viel Zeit zur Erkennung und Analyse von Vorfällen vergeht, bis diese effektiv behoben werden können. Um die Reaktivität zu minimieren, sollten zum einen potentielle Bedrohungen im Vorfeld schnellstmöglich erkannt werden. Zum anderen sollten Vorfälle priorisiert werden.

Der Einsatz von Cyber Threat Intelligence kann den genannten Herausforderungen, wie auch schon in Abschnitt 3.1 kurz erwähnt, entgegenwirken. CTI bietet aktuelle Informationen über bekannte TTPs, spezifische Angriffsmethoden sowie Erkenntnisse über etwaige Datenleaks. Es können externe Informationen mit den vorhandenen internen Informationen verglichen werden, um konkrete Bedrohungen schneller zu identifizieren. Weiterhin stellt Pace u. a. [72] vier Merkmale zur Charakterisierung von CTI vor. Sie sollten umfassend sein und automatisiert erfasst werden, um manuelle Recherchen und somit weiteren Zeitverlust zu unterbinden. Weiterhin müssen CTI für das jeweilige IRT von Bedeutung sein. CTI sollte zudem einen gewissen Kontext mitbringen, damit wichtige Hinweise erkannt werden. Beispielsweise kann der zeitliche Verlauf eines Vorfalls aufzeigen, welche weiteren Ereignisse aufgetreten sind, bevor es zu dem Vorfall kommen konnte. Zuletzt soll CTI für unterschiedliche Threat Intelligence Plattformen integrierbar sein, damit *false positives* automatisiert erkannt und aussortiert werden können. Dies erspart eine manuelle Bearbeitung und folglich viel Zeit und Kosten.

3.3 IT-Forensics-Team

Ein IT-Forensics-Team ist eine weitere Zielgruppe von CTI und kann zudem ein Bestandteil eines Incident Response Team oder Security Operations Center sein. Es befasst sich im Kern mit der sachlichen Aufklärung von Sicherheitsvofällen. Dadurch kann es als Brücke zwischen der Reaktion auf einen Vorfall und einer möglichen Strafverfolgung angesehen werden. Somit ergeben sich diverse Ziele an eine forensische Untersuchung. Es muss untersucht werden, was genau vorgefallen ist und welche Systeme betroffen sind. Außerdem muss ermittelt werden, wie der Vorfall zustande gekommen und wann dieser aufgetreten ist. Ferner kann analysiert werden, wie zukünftige Vorfälle dieser Art verhindert werden können. Zur weiteren Strafverfolgung ist es zudem sinnvoll zu erkennen, wer

für den Vorfall verantwortlich ist. [2]

In der IT-Forensik existiert nach Kent u. a. [30] ein grundlegender Prozess, um Sicherheitsvorfällen nachzugehen. In der ersten Phase werden Daten gesammelt, um diese als mögliche Hinweise eines Vorfalls zu verwenden. Dies können Logeinträge von Netzwerkaktivitäten oder Anwendungen sein. Besonderes Augenmerk liegt dabei auf die weitere legale Verwendung der Daten als Beweismittel für zukünftige mögliche Strafverfahren. Die zweite Phase sieht eine Überprüfung der gesammelten Daten vor. Es können mitunter viele Daten gesammelt werden, die viele Informationen enthalten und nicht zwangsläufig relevant sein müssen. Daher bedarf es einer Filterung dieser, um die wichtigsten Informationen zu erhalten. Innerhalb der dritten Phase werden die extrahierten Informationen analysiert. Die Analyse soll, im Kontext von illegalen Aktivitäten, idealerweise Aufschluss zu dem Aufenthaltsort oder der Identität eines Angreifenden geben. Häufig ist es notwendig Informationen aus weiteren Quellen hinzuzuziehen und diese miteinander zu korrelieren. Dies können weitere Logeinträge eines Network Intrusion Detection Systems (NIDS) sein, wodurch IP-Adressen der Angreifenden ermittelt werden können. Zuletzt werden die analysierten Informationen für eine jeweilige Zielgruppe verständlich zusammengefasst. Werden die Informationen an eine Law Enforcement Agency weitergeleitet, bedarf es einer detaillierten und korrekten Zusammenfassung des Vorfalls. Entscheidungsträger*innen erwarten hingegen eine abstraktere Präsentation der gewonnenen Erkenntnisse zu einem Vorfall z.B. mithilfe von visuellen Darstellungen. Systemadministrator*innen benötigen hingegen technische Details.

Dieser Prozess deutet bereits daraufhin, dass CTI als Quelle genutzt werden kann und zudem neue CTI, z.B. Strategic Threat Intelligence und Technical Threat Intelligence, produziert werden. Zudem zeigt Serketzis u. a. [82], dass mithilfe von CTI und dem Einsatz von Threat Intelligence Platforms (TIPs) die Wirksamkeit innerhalb der IT-Forensik gesteigert werden kann.

3.4 Law Enforcement Authorities

Während die Internetkriminalität in den letzten Jahren weiter angestiegen ist, nutzen Law Enforcement Authorities (Strafverfolgungsbehörden) die Techniken der IT-Forensik zur Aufklärung von Delikten. Somit fallen sie ebenfalls in die Zielgruppe von CTI. Studien zeigen hingegen, dass die Behörden wenig oder schlecht ausgebildetes Personal haben die sich mit der Forensik von Vorfällen befassen [1]. Zusätzlich mangelt es an Software für die

Behörden zur Analyse von Vorfällen [22]. Dempsey [10] zeigt jedoch auf, wie vorteilhaft es sein kann, wenn Law Enforcement Authorities im Besitz von Bedrohungsinformationen sind. Ist ein Unternehmen beispielsweise von einem Vorfall betroffen, können sie sich an die Behörden wenden, um bereits existierendes Wissen von ähnlichen oder gleichen Vorfällen einzuholen. Das Wissen können diese sodann durch eigene forensische Untersuchungen erlangt oder durch gemeldete Vorfälle erhalten haben. Dies kann maßgeblich zur weiteren Aufklärung dienen. Eine Vernetzung der Behörden über Landesgrenzen hinaus ist zudem von Vorteil, um international agierende Hackergruppen effizienter aufdecken zu können.

Da keine konkreten Quellen vorhanden sind, die die Verwendung oder den Nutzen von CTI für Law Enforcement Authorities aufzeigen, kann an dieser Stelle nur geschlussfolgert werden, inwieweit es diesbezüglich Zusammenhänge gibt. Da aus dem vorangegangenen Unterkapitel bereits deutlich wurde, dass IT-Forensics-Teams aus den Analyseergebnissen eines Sicherheitsvorfalls u.a. Zusammenfassungen für Strafverfolgungsbehörden erstellen, kann hier eine Verknüpfung zwischen CTI und den Behörden hergestellt werden. Des Weiteren erstellen die Behörden eigene Bedrohungsinformationen, wie aus der privaten Kommunikation mit dem betreuenden Prüfer Prof. Dr. Klaus-Peter Kossakowski hervorgeht. Allerdings ist der Austausch schwierig.

3.5 Fraud Prevention Team

Die Zielgruppe Fraud Prevention Team (Betrugspräventionsteam) nutzt CTI zum Aufzeigen von Strukturen organisierter Internetkriminalität und konkreter Bedrohungen sowie Vorgehensweisen. Die in diesem Bereich operierenden kriminellen Gruppen sind mit einem sogenannten Mastermind an der Spitze hierarchisch strukturiert. Ferner besitzen sie ausgebildete Mitglieder aus verschiedenen Bereichen, wie z.B. dem Finanzsektor, den Law Enforcement Authorities oder der Softwareentwicklung. Diese Personen haben ein hohes Ansehen in der Gesellschaft, sodass niemand darauf schließen würde, dass sie kriminelle Handlungen durchführen. Die Kommunikation von organisierten kriminellen Gruppen findet in privaten Foren oder über bekannte Nachrichtendienste wie Jabber oder Telegram statt. Ihre Dienstleistungen werden i.d.R. kostenpflichtig in Foren oder in Marketplaces im Deep oder Dark Web angeboten. [72]

Durch die Beobachtung der Foren, Marketplaces sowie Nachrichtenkanäle, können diverse Bedrohungsinformationen gesammelt werden. Es ist dadurch möglich, die Taktiken

sowie Vorgehensweisen krimineller Gruppierungen aufzuzeigen oder eingesetzte Schadprogramme zu identifizieren. Ferner werden entwendete sensible Informationen, wie Kreditkartendaten oder Zugangsdaten u.a. für bezahlpflichtige Dienste, durch diese Gruppen zugänglich gemacht. Dadurch ist eine missbräuchliche Einsicht und Nutzung problemlos möglich. Überdies können diese Informationen von Strafverfolgungsbehörden genutzt und mit weiteren Indikatoren korreliert werden. Mithilfe der Informationen aus verschiedenen Quellen können Verbindungen von konkreten Vorfällen zu kriminellen Gruppen aufgezeigt und bewiesen werden. [72]

Auf der einen Seite ist erkennbar, dass Technical Threat Intelligence, in Form von Kreditkartendaten oder Hashwerte eingesetzter Schadprogramme, vorhanden sind. Auf der anderen Seite können Tactical Threat Intelligence, u.a. zur Identifizierung von Vorgehensweisen, erschlossen werden. Um international agierende Gruppierungen zu verfolgen, ist es sinnvoll die Informationen zu teilen, auch wenn dies in der Praxis schwierig ist. Daher konsumieren Behörden die gegen Betrugsdelikte vorgehen zum einen CTI und zum anderen produzieren sie diese.

3.6 Chief Information Security Officer

Der mittlerweile größte Verantwortungsbereich eines Chief Information Security Officer (CISO) ist das Risiko Management, welcher im folgenden Abschnitt beleuchtet wird. Ein CISO muss mit den vorhandenen Ressourcen und finanziellen Mitteln einen Weg finden, Bedrohungen gegen das eigene Unternehmen zu ermitteln und effizient abzuschwächen. Zur Risikobewertung und Erstellung von Sicherheitsstrategien müssen zunächst die Sicherheitsanforderungen ermittelt werden. Anschließend wird ermittelt, welche Sicherheitsvorkehrungen bereits vorhanden sind. Darauf aufbauend werden die wesentlichsten Lücken im Sicherheitskonzept benannt, priorisiert und Maßnahmen zur Lösung ermittelt. Zuletzt werden die Maßnahmen beobachtet und evaluiert. Mit geeigneten Metriken kann die Effektivität bewertet werden. Eine weitere Aufgabe für einen CISO besteht in der Kommunikation mit der Unternehmensführung. Ihnen gegenüber sind die getätigten Investitionen im Risiko Management zu rechtfertigen und offen zu legen. [72]

Um den Aufgaben gerecht zu werden, ist es notwendig, ausreichend Informationen zu relevanten Risiken sowie Bedrohungen für das Unternehmen zu besitzen. Die bereits intern gewonnenen Informationen bspw. aus früheren Vorfällen oder aufgezeichneten System-Logs sind nicht ausreichend, um geeignete Sicherheitsstrategien zu entwickeln. Der Bezug

von CTI aus externen Quellen, idealerweise von Unternehmen aus dem gleichen wirtschaftlichen Sektor, liefert den notwendigen Kontext. Aus den Informationen kann zum einen ermittelt werden, welche Art von Angriffen zur Zeit im Vordergrund stehen. Zum anderen kann ermittelt werden, welche Vorgehensweisen zur Verhinderung von Angriffen notwendig und effektiv sind. Um eine Übersicht über die große Menge vorhandener Informationen zu behalten und, um die notwendigsten Informationen herauszufiltern, werden die bereits eingeführten operativen Sicherheitsteams (wie das Security Operations Center oder Incident Response Teams) eingesetzt. [72]

Unternehmen sind u.a. durch technische Schwachstellen, durch Insider-Angriffe oder durch Techniken des Social Engineering angreifbar. CTI bietet Informationen die Aufschluss über eingesetzte TTPs der Angreifenden offen legen, um zu erkennen welche Schwächen des Unternehmens ausgenutzt werden. Mithilfe von CTI kann ergänzend ein für das Unternehmen spezifische Risikoprofil erstellt werden. Dieses Profil wird u.a. durch die Branche, den Unternehmensstandort sowie der internen Infrastruktur gebildet. Somit können die Bereiche aufgezeigt werden, in die notwendige Investitionen zum Schutz vor Angriffen, basierend auf erfolgreiche Angriffe auf vergleichbare Unternehmen, vorzunehmen sind. [72]

Zu erkennen ist, dass ein CISO eines Unternehmens auf verschiedene Cyber Threat Intelligence angewiesen ist. Zum einen kann ein Unternehmen eigene Tactical und Technical Threat Intelligence besitzen. Zum anderen bezieht es die gleichen Arten von Informationen aus weiteren Quellen, welche im Idealfall aus dem gleichen wirtschaftlichen Bereich kommen sollten. Gleichzeitig werden sie zudem an andere Unternehmen verteilt. Letztlich ist das Ziel der Unternehmensführung die Risikolage zu schildern, um etwaige Investitionen für einen Unternehmensschutz zu rechtfertigen. Daraus kann abgeleitet werden, dass die Darstellung der Ergebnisse mithilfe von Berichten oder visuellen Mitteln erfolgt. Somit produziert diese Zielgruppe zusätzlich Strategic Threat Intelligence.

3.7 Risk Manager

Eine, durch einen Risk Manager (Risikomanager) durchgeführte, Risikomodellierung bietet eine objektive Beurteilung von Risiken sowie eine qualitative und quantitative Abschätzung zur Investition in die Sicherheit eines Unternehmens. Das Risiko wird hierbei aus dem Produkt von Eintrittswahrscheinlichkeit eines Vorfalls und den Auswirkungen der direkten und indirekten Folgen gebildet. Mithilfe etablierter Risikomodelle, wie dem

Factor Analysis of Information Risk (FAIR) Modell [90], werden quantifizierte Bewertungen von Risiken gebildet. Diese beinhalten zudem die Wahrscheinlichkeiten für den zu erwartenden Verlust in Abhängigkeit der jeweiligen Bedrohung. Bedrohungen sind z.B. Phishing- oder Ransomware-Angriffe. Ein Vorteil des FAIR-Modells ist die Transparenz der ermittelten Werte, sodass alle Beteiligten diese einsehen können. Weiterhin ist dadurch für alle Involvierten eine gemeinsame verständliche Sprache gegeben. Alle Beteiligten besitzen hierdurch eine Grundlage über die diskutiert und letztlich ein besseres Verständnis und damit größeres Vertrauen in die Ergebnisse aufgebaut werden kann. [72]

Der Einsatz von CTI kann bei der Risikomodellierung verschiedene Fragen klären [72]:

- Welche Threat Actors nutzen welche spezifischen Angriffe?
- Betreffen sie zudem die eigene Branche?
- Wie verhält sich der Trend zu den Angriffen?
- Welche Schwachstellen wurden ausgenutzt?
- Wie hoch ist der finanzielle sowie technische Schaden, die ähnliche Unternehmen erfahren haben?

Pace u. a. [72] machen nicht deutlich, welche Arten von CTI notwendig sind, um die obigen Fragen zu klären, bzw. um eine adäquate Risikomodellierung durchführen zu können. Allerdings kann dem Leitfaden entnommen werden, dass z.B. ein zeitliches Verhalten einer Ransomwarefamilie anhand verschiedener Indikatoren aufgezeigt werden kann. Diese werden aus unterschiedlichen Quellen, wie bspw. aus Untergrundforen oder Code Repositories, bezogen und miteinander in Verbindung gebracht. Diese Informationen zählen zu den Technical Threat Intelligence. Weiterhin ist der Bezug von Tactical Threat Intelligence aufschlussreich, um durch die Identifizierung von Angriffsmustern zukünftige Angriffe in ihrem Risiko besser bewerten zu können. Dadurch kann ein vorausschauender Schutz dagegen aufgebaut werden. Die Risikomodellierung führt eine Beurteilung von Risiken durch. Strategic Threat Intelligence verfolgt u.a. das Ziel neue Risiken zu erkennen und bestehende besser zu verstehen. Daraus kann gefolgert werden, dass durch den Risk Manager jene Bedrohungsinformationen erzeugt werden, damit die Entscheidungsträger*innen diese verwenden können.

3.8 Vulnerability Management Team

Vulnerability Management Teams befassen sich mit der proaktiven Reduzierung möglicher Schwächen von Systemen. Hierbei ist zwischen Vulnerability Management Teams eines Herstellers und solchen von Betreibern eines Systems zu unterscheiden. Dabei handelt es sich weniger um das Beseitigen aller Schwachstellen. Es sollen vielmehr ausgewählte Schwachstellen behoben werden, die tatsächlich ein eingesetztes System betreffen. Pace u. a. [72] macht deutlich, dass viel mehr Schwachstellen vorhanden sind, als tatsächlich ausgenutzt werden. Weiterhin werden tendenziell mehr Exploits für Schwachstellen entwickelt, die Technologien betreffen, die weit verbreitet sind, wie z.B. Windows-Systeme. Aufgrund dieser Herleitung sollten Schwachstellen erkannt und beseitigt werden, die ein Unternehmen am wahrscheinlichsten betreffen können. [72]

Schwachstellen können standardisiert durch Common Vulnerabilities and Exposures (CVE) beschrieben und mit dem Common Vulnerability Scoring System (CVSS) in ihrer Schwere bewertet werden. Datenbanken zur Dokumentation von Schwachstellen, wie z.B. die National Vulnerability Database (NVD) [55] vom National Institute of Standards and Technology , haben i.d.R. zwei Nachteile. Auf der einen Seite dokumentieren sie vorhandene Schwachstellen, allerdings ohne dem Hinweis, ob sie wirklich ausgenutzt werden. Exemplarisch ist dies an der Schwachstelle CVE-2017-0022 [54] zu erkennen, deren CVSS bei 4,3 (medium) von 10 (critical) liegt. Eine genauere Beschreibung von CVSS wird in Abschnitt 6.5 vorgenommen. Allerdings ist das reale Risiko der Ausnutzung dieser Schwachstelle laut Pace u. a. [72] deutlich höher, da bereits ein Exploit in einem bekannten Exploit-Kit integriert wurde. Auf der anderen Seite werden neue Schwachstellen nicht schnell genug verzeichnet. Typischerweise wird eine Schwachstelle den Verantwortlichen der NVD gemeldet, welche sie aber nicht direkt veröffentlichen. Gleichzeitig können weitere Informationen über die Schwachstelle bspw. über die sozialen Medien durch den Entdecker oder der Entdeckerin veröffentlicht werden. Dies birgt die Gefahr, dass ein Schadprogramm zur Ausnutzung der Schwachstelle entwickelt und eingesetzt werden kann. [72]

Diese Datenbanken dürfen darum nicht die einzige Quelle von CTI bleiben. Zum Beispiel können Blogs von Sicherheitsforscher*innen, soziale Medien oder diverse Repositories zur Veröffentlichung von PoCs auf GitHub herangezogen werden, um detailliertere Informationen zu gewinnen. Weiterhin müssen die eigenen Systeme vom Vulnerability Management Team untersucht werden, um Informationen über existierende Schwachstellen zu

erhalten. Durch Korrelation dieser Daten, kann ein Gesamtbild zur Risikobewertung einer spezifischen Schwachstelle erstellt und das weitere Vorgehen evaluiert werden. Ein ideales Vorgehen sieht vor, dass die Schwachstellen durch die zuständige IT-Abteilung geschlossen werden. [72]

Es wurde deutlich, dass diese Zielgruppe auf der einen Seite Technical Threat Intelligence zur Identifizierung von Schwachstellen innerhalb der eigenen Infrastruktur einsetzt. Auf der anderen Seite werden durch die Nutzung externer Quellen zudem Tactical Threat Intelligence eingesetzt. Diese dienen der Priorisierung der Notwendigkeit zur Beseitigung von Schwachstellen. Ob weitere Bedrohungsinformationen produziert werden, um z.B. die Unternehmensführung über ermittelte Risiken zu informieren, ist indes nicht erkennbar.

3.9 Governments

Eine weitere Zielgruppe von CTI stellen Governments (Regierungen) dar. Wie sie Bedrohungsinformationen einsetzen und welche Ziele sie damit verfolgen wird im folgenden exemplarisch an der „National Intelligence Strategy of the United States of America“ [69] gezeigt. Aus diesem Strategiepapier geht hervor, wie die US-Regierung auf strategischer Ebene Cyber Threat Intelligence einsetzen will. Allgemeine Ziele sind u.a. die Wahrung der nationalen Sicherheit gegenüber Terrorismus oder den Schutz vor Angriffen auf kritische Infrastrukturen. Damit diese Ziele erreicht werden können wurde eine sogenannte Intelligence Community (IC) aufgebaut. Darin sind u.a. die National Security Agency (NSA) und Central Intelligence Agency (CIA) vertreten. Ihre Aufgaben bestehen hauptsächlich darin Informationen zu sammeln, diese zu analysieren und die gewonnenen Erkenntnisse für verschiedene Empfänger*innen bereitzustellen. Dies sind beispielsweise der US-Präsident oder die US-Präsidentin sowie Law Enforcement Authorities oder der nationale Sicherheitsrat der USA.

Mithilfe von CTI sollen Bedrohungen erkannt und nachvollzogen werden können. Die gewonnenen Erkenntnisse sollen, wie schon erwähnt, an Verantwortliche zur Entscheidungsfindung weitergeleitet werden. In der Veröffentlichung wird CTI als ein Prozess definiert, welcher Informationen aus diversen Quellen, die das Interesse der nationalen Sicherheit betreffen, sammelt, verarbeitet, analysiert sowie weiterleitet. Die Informationen umfassen u.a. Taktiken von Angreifenden oder Indikatoren für mögliche Vorfälle. Ferner können Erkenntnisse über potentielle Auswirkungen eines Angriffs oder die für

einen Angriff eingesetzte Infrastruktur gewonnen werden. Es wird ebenfalls deutlich, dass das Bewusstsein für Bedrohungen und der Schutz davor weiter wächst, jedoch die Risiken in vielen Bereichen weiterhin bestehen. Überdies sollen die gesammelten Informationen durch Partnerschaften untereinander ausgetauscht werden. Dazu gehören u.a. das US-Militär, private Firmen oder ausländische Geheimdienste. Um dies zu gewährleisten bedarf es an geeigneten Technologien für den Datenaustausch. [69]

Die Arten von CTI, die zur Präsentation der gewonnenen Erkenntnisse aus der Analyse gesammelter Informationen genutzt werden, ist indes nicht explizit genannt. Es lässt sich jedoch erkennen, dass sie u.a. als Strategic Threat Intelligence zu sehen sind, da die Zielgruppen u.a. Entscheidungsträger*innen beinhalten. Des Weiteren ist denkbar, dass Operational Threat Intelligence durch einige der IC-Mitglieder erzeugt und konsumiert werden können. Beispielsweise ist auf der einen Seite nicht auszuschließen, dass die NSA mit dem Einsatz vieler ihrer Überwachungsprogramme, bevorstehende Angriffe durch die Analyse von CTI identifizieren können und die Erkenntnisse den Law Enforcement Authorities weitegeben. Auf der anderen Seite kann die NSA Operational TI beziehen, um bevorstehende Angriffe abzuwehren. Darüber hinaus wird deutlich, dass sowohl Technical als auch Tactical Threat Intelligence gesammelt und weitergeleitet werden. Diese können anderen Behörden oder privaten Unternehmen zur Verfügung gestellt werden, sodass sie selbstständig vorbeugende Maßnahmen für einen Schutz vor Angriffen treffen können.

4 Forschungsfragen und Anforderungen

4.1 Forschungsfragen und Hypothesen

Einleitend wurde bereits der stetige Anstieg von Malware in den vergangenen Jahren aufgezeigt. Außerdem ist die Anzahl der Schwachstellen in der National Vulnerability Database (NVD) in den letzten Jahren schrittweise gestiegen wie die Kurve in Abbildung 6.1 auf Seite 87 erkennen lässt. In dem Bericht „Is Software More Vulnerable Today? “ [14] der ENISA, welcher auf die selbe Statistik der NVD aus dem Jahr 2018 zurückgreift, werden verschiedene Gründe für den Anstieg genannt. Zum Beispiel werden Qualitätssicherung und -prüfung für Soft- und Hardware vernachlässigt, wodurch Schwachstellen nicht entdeckt werden. Ebenso versuchen Kriminelle verstärkt Schwachstellen für den unberechtigten Zugang zu Systemen zu finden. Das Wissen darüber oder bereits entwickelte Exploits zur Ausnutzung der Schwachstellen können für viel Geld verkauft werden.

Schadprogramme können einen finanziellen Schaden für Unternehmen verursachen. Zudem können Angreifende dadurch persönliche Daten beliebiger Personen stehlen oder gezielt missbrauchen. Daher besteht ein großer Bedarf mithilfe von CTI bspw. mögliche Indikatoren für einen Vorfall oder Strategien der Angreifenden zu dokumentieren. Dadurch ist es für die Zielgruppen von CTI möglich, schnell und effizient Angriffe zu erkennen bzw. darauf zu reagieren, Entscheidungen zur Reduzierung weiterer Folgen und Schäden zu treffen oder Auswirkungen zu minimieren.

Es gibt diverse Quellen, Datenformate, Protokolle und Plattformen die dafür genutzt werden können. Einige wurden bereits in den Abschnitten 2.3 und 2.4 vorgestellt. Eine Übersicht bieten zum Beispiel die European Union Agency For Network and Information Security (ENISA) [15], Tounsi und Rais [92], Kampanakis [29] oder Slatman [87]. Demzufolge existieren verschiedene Infrastrukturen und Möglichkeiten zur Verarbeitung dieser Informationen. Darauf aufbauend können weitere Quellen zur Erzeugung von CTI erschlossen werden. Mit dieser Arbeit soll dies anhand einer Analyse von Exploits aus

der ExploitDB erreicht werden (siehe Abschnitt 2.4). Dies bedingt zunächst eine konkretere Einordnung des Exploit-Begriffes, die Benennung von Forschungsfragen sowie einige daraus resultierende Hypothesen.

Ein Exploit ist nach dem Bundesamt für Sicherheit in der Informationstechnik (BSI) [3] zunächst ein Schadprogramm zur Ausnutzung von Schwachstellen in Computerprogrammen. Diese Definition ist allerdings zu eng gefasst. Ein Exploit kann allgemeiner definiert werden. Es stellt vielmehr einen allgemeinen Prozess zur Ausnutzung jeder Art von Schwachstellen dar. Nach einer Studie des Information Security Forum (ISF), welche von Sheridan [85] zusammengefasst wurde, können zudem psychologische Schwachstellen eines Menschen ausgenutzt werden, z.B. mithilfe von Social Engineering.

In Abschnitt 3.2 wurden die Begriffe *precursor* und *indicator* als Kategorien für Hinweise zur Identifizierung von möglichen Vorfällen angesprochen. Hinweise in der *precursor*-Kategorie deuten auf einen möglichen Vorfall in der Zukunft hin, wohingegen Hinweise aus der *indicator*-Kategorie auf einen bereits eingetretenen oder gerade auftretenden Vorfall hindeuten. Die aus der ExploitDB betrachteten Exploits sind typische Quellen zur Ausnutzung von technischen Schwachstellen. Aus ihnen sollen in erster Linie die zuvor erwähnten Hinweise zur Erkennung von Vorfällen extrahiert werden.

Daraus ergeben sich die Forschungsfragen und Hypothesen in Tabelle 4.1.

Nummer	Forschungsfrage	Hypothese(n)
FF01	Welche Hinweise zur Identifizierung von Vorfällen können aus den Exploits der ExploitDB extrahiert werden?	Wenn Exploits Quellen für Hinweise aus den Kategorien *precursor* sowie *indicator* sind, dann können aus den Exploits der ExploitDB ebenfalls Hinweise aus diesen Kategorien extrahiert werden.
FF02	In welche der vier Kategorien von CTI (Strategic Threat Intelligence, Operational Threat Intelligence, Tactical Threat Intelligence, Technical Threat Intelligence), nach Chismon und Ruks [4] aus Abschnitt 2.1, können die gewonnenen Informationen eingegliedert werden?	Wenn Hinweise aus der *precursor*- und *indicator*-Kategorie aus diesen Exploits gewonnen werden können, dann gehören sie zu den Technical Threat Intelligence. Wenn die gewonnenen Informationen Aufschluss über das Vorgehen von Angreifenden geben, dann zählen sie zu den Tactical Threat Intelligence.
FF03	Welche Zielgruppen können die gewonnenen Informationen aus den Exploits verwenden?	Wenn die Informationen als CTI zu betrachten sind (siehe FF02), dann können alle in Kapitel 3 genannten Zielgruppen diese für die eigenen Ziele nutzen.
FF04	Welche Aussage kann zwischen der Anzahl veröffentlichter Schwachstellen und der Anzahl veröffentlichter Exploits getroffen werden?	Wenn es eine bestimmte Anzahl veröffentlichter Schwachstellen pro Jahr gibt, dann gibt es, relativ gesehen dazu, viele veröffentlichte Exploits im selben Jahr.
FF05	Kann ein zeitlicher Zusammenhang zwischen der Veröffentlichung einer neuen CVE und einem Exploit hergestellt werden?	Wenn eine neue CVE veröffentlicht wurde, dann folgt kurze Zeit später die Veröffentlichung eines dazugehörigen Exploits.

Nummer	Forschungsfrage	Hypothese(n)
FF06	Für welche Plattform stehen die meisten Exploits zur Verfügung?	Je populärer eine Plattform ist, desto eher ist sie Ziel von Angriffen durch Exploits.
FF07	In welcher Programmiersprache werden die meisten Exploits verfasst?	Je populärer eine Programmiersprache ist, desto eher wird diese zur Erstellung von Exploits eingesetzt.

Tabelle 4.1
Forschungsfragen und Hypothesen

4.2 Eingliederung in den CTI-Lifecycle

Der Ablauf zur Analyse von Python-Exploits, des hier vorzustellenden Prototyps, kann in dem Großteil des CTI-Lifecycles aus Abschnitt 2.2 wiedererkannt werden. Die Phasen *Direction* sowie *Feedback* werden nicht abgedeckt, da es sich um ein Hochschulprojekt handelt und kein Unternehmen involviert ist. Ein grober Ablauf der Analysesoftware sieht zunächst das Sammeln von Daten (*Collection*-Phase) vor. Anschließend werden sie in ein einheitliches Format transformiert (*Processing*-Phase). Die Verwendung unterschiedlicher Quellen ist jedoch nicht vorgesehen. Danach werden die Daten analysiert (*Analysis*-Phase). Zuletzt werden die Erkenntnisse der Analyse in einem bestimmten Format innerhalb von MISP zur Verfügung gestellt (*Dissemination*-Phase). Die Beschreibungen der funktionalen Anforderungen im folgenden Abschnitt, zeigen einen detaillierteren Ablauf der einzelnen Schritte auf.

4.3 Funktionale Anforderungen

Tabelle 4.3 auf Seite 54 listet die funktionalen Anforderungen an die Software für eine Exploit-Analyse auf. Sie werden durch eine eindeutige Nummer, einer kurzen Bezeichnung und einer Beschreibung spezifiziert. Diese werden im Folgenden genauer erörtert.

Die offizielle Webseite „Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers“ [63] bietet keine Schnittstelle zum Herunterladen der Exploits an. Allerdings werden die Exploits in einem GitHub Repository [68] öffentlich zugänglich

bereitgestellt. Aus diesem Grund soll es eine Komponente in dem System geben, die das Repository herunterlädt und die Daten lokal abspeichert (FA01). Zusätzlich soll diese Komponente die Möglichkeit besitzen, das lokale Repository regelmäßig zu aktualisieren, da neue Exploits kontinuierlich eingepflegt oder existierende aktualisiert werden.

In dem Repository steht eine CSV-Datei zur Verfügung, in der bereits einige Metadaten über jeden einzelnen Exploit vorzufinden sind. Daher soll das System diese Informationen extrahieren und in eine für die Analyseergebnisse vorgesehene Datenbank speichern (FA02). Tabelle 4.2 führt das jeweilige Metadatum auf, beschreibt es kurz und gibt die Werte eines bestimmten Exploits exemplarisch als ein Beispiel aus der Datei an [67].

Metadatum	**Beschreibung**	**Beispiel**
id	Eine eindeutige ID eines Exploits.	46780
file	Der relative Pfad und die Bezeichnung zu einem Exploit in dem Repository.	exploits/windows/webapps/46780.py
description	Eine kurze Beschreibung des Exploits.	„Oracle Weblogic 10.3.6.0.0 / 12.1.3.0.0 - Remote Code Execution“
date	Das Datum der Veröffentlichung des Exploits.	2019-04-30
author	Der Name des/der Autors/Autorin des Exploits.	„Avinash Kumar Thapa“
type	Einer der vier Typen eines Exploits: DoS, Local, Remote oder Webapps.	webapps
platform	Die Plattform für den der Exploit entwickelt wurde: Windows, Linux, etc..	windows
port	Optional ein Port, welcher für die jeweilige Ausnutzung einer Schwachstelle genutzt wird.	-

Tabelle 4.2
Aufbau der CSV-Metadaten eines Exploits mit einem realen Beispiel

Wie in der Einleitung formuliert, soll für die Erkennung etwaiger Trends die Möglichkeit zur Generierung von Statistiken vorhanden sein. Dafür sollen die Daten aus der Daten-

bank ausgelesen, die Statistiken berechnet und diese jeweils als Grafik auf der Festplatte abgespeichert werden (FA03).

Die Exploits stehen in einer Datei mit der jeweiligen Endung der verwendeten Programmiersprache zur Verfügung. Sie setzen sich zum einen aus Kommentaren und zum anderen dem Quellcode zusammen. Für eine klare sowie modulare Trennung in der Analyse zwischen Kommentaren und Quellcode, sollen diese voneinander getrennt und separat abgespeichert werden (FA04).

Zur weiteren Erzeugung von Metadaten bzw. möglichen Indikatoren zu einem Exploit, sollen die separierten Kommentare analysiert werden (FA05). Dies soll durch ein Pattern- und String-Matching durchgeführt werden. Daher muss im Vorfeld die Möglichkeit bestehen die Pattern und Zeichenketten (Strings) zu definieren. Die Ergebnisse werden sodann in die Datenbank zu den bereits existierenden Informationen eines Exploits abgespeichert.

Weitere Informationen oder Indikatoren zu einem Exploit sollen durch eine Codeanalyse gewonnen werden (FA06). Hierzu soll ein Syntaxbaum des jeweiligen Exploits aufgebaut und analysiert werden. Die hierbei gewonnenen Erkenntnisse reichern die bestehenden Informationen in der Datenbank an.

Nachdem alle Exploits analysiert und die gewonnenen Informationen in die Datenbank abgespeichert wurden, sollen diese zur Erstellung neuer oder zur Anreicherung bestehender Events in MISP eingesetzt werden (FA07). Daher ist es notwendig, dass eine Verbindung zu einer MISP-Instanz besteht. MISP besitzt ein REST-API, welches hierfür genutzt werden soll.

Nummer	Bezeichnung	Beschreibung
FA01	Herunterladen von Exploits	Das System soll die Exploits aus dem GitHub Repository von Offensive Security [68] herunterladen und auf der Festplatte speichern.
FA02	Parsen und Speichern der Exploit Metadaten	Das System soll die vorhandenen Metadaten die in einer CSV-Datei vorliegen parsen und in eine Datenbank speichern.
FA03	Erstellen von Statistiken	Das System soll visuelle Statistiken (Aggregationen, Zeitreihen, etc.) erzeugen und als Grafiken auf der Festplatte speichern.
FA04	Exploits parsen	Das System soll die vorhandenen Exploits parsen. Das heißt, der konkrete Exploitcode muss von den Kommentaren in der Exploitdatei unterschieden werden.
FA05	Extrahieren von weiteren Metadaten	Das System soll aus den Kommentaren, die in *FA04* extrahiert wurden, weitere Metadaten generieren und die Daten in der Datenbank damit anreichern.
FA06	Analyse des Exploitcodes	Das System soll eine Codeanalyse der Exploitcodes aus *FA04* durchführen und die Daten in der Datenbank mit den gewonnenen Informationen anreichern.
FA07	MISP Integration	Das System soll Bedrohungsinformationen aus den gewonnenen Informationen der Analyse neu erstellen und nach MISP exportieren oder bereits bestehende Einträge in MISP damit anreichern.

Tabelle 4.3
Funktionale Anforderungen

4.4 Nicht-Funktionale Anforderungen

Tabelle 4.4 auf Seite 56 listet die nicht-funktionalen Anforderungen an die Software für eine Exploit-Analyse auf. Sie werden durch eine eindeutige Nummer, einer kurzen Bezeichnung und einer Beschreibung spezifiziert. Diese werden im Folgenden kurz beschrieben.

Durch einen modularen Aufbau ist gewährleistet, dass weitere Komponenten mit wenig Aufwand integriert werden können. Denkbar ist u.a. eine weitere Quelle für Exploits oder weitere Senken als MISP hinzuzufügen (NFA01).

Eine Verarbeitung und Bereitstellung von CTI in Echtzeit ist häufig eine Anforderung [73]. Für eine schnelle Reaktion auf mögliche Vorfälle in naher Zukunft oder jene die bereits vorgefallen sind, benötigt es eine schnelle Bereitstellung von Informationen (NFA02).

Nicht korrekte Informationen (*false positives*) sind häufig ein Problem, da diese zu einem Fehlalarm führen können [73]. Dadurch wäre es beispielsweise möglich, dass ein Exploit als sehr kritisch definiert wird, obwohl er dies tatsächlich nicht ist. Diese sollten vermieden werden, damit die Ergebnisse nicht verfälscht werden und somit keine falschen Entscheidungen getroffen werden oder zu voreilig reagiert wird (NFA03).

Alle Exploits sollen auf die gleiche Art und Weise analysiert werden. Es ist aber nicht zwangsläufig gegeben, dass jeder Exploit die gleichen Metadaten enthält. Insofern wird zur Speicherung dieser Daten eine tolerante Datenbank benötigt (NFA04).

Zur Vorbeugung falscher Informationen (s. NFA03) sowie einer korrekten Funktionsweise, sollen alle Komponenten mit Tests überprüft werden können (NFA05).

Nummer	Bezeichnung	Beschreibung
NFA01	Änderbarkeit	Das System soll modular aufgebaut werden, damit der Code wartbar ist und weitere Komponenten problemlos angegliedert werden können.
NFA02	Zeitverhalten	Das System soll in allen Bereichen der Datenverarbeitung (Parsen, Speichern, Codeanalyse, etc.) effizient arbeiten, damit Bedrohungsinformationen zügig erstellt oder mit weiteren Informationen angereichert werden können.
NFA03	Korrektheit	Das System soll korrekte Metadaten aus der Analyse erzeugen.
NFA04	Fehlertoleranz	Das System soll bei der Anreicherung der Daten insofern fehlertolerant sein, sodass einzelne Datensätze unterschiedliche Attribute aufweisen können.
NFA05	Prüfbarkeit	Die Funktionsweisen einzelner Komponenten des Systems sollen durch Tests überprüfbar sein.

Tabelle 4.4
Nicht-Funktionale Anforderungen

5 Softwaredesign und technische Realisierung

Die folgenden Abschnitte stellen das Design und die technische Realisierung der ExploitDB-Analysesoftware vor. Zunächst wird die grundlegende Architektur visuell aufgezeigt. Anschließend wird auf die jeweiligen Softwarekomponenten eingegangen. Zuletzt wird das Datenmodell erläutert. Daran anknüpfend wird die technische Realisierung erörtert. Dabei werden die eingesetzten technischen Hilfsmittel zur Umsetzung des Designs beschrieben sowie diverse Besonderheiten die bei der Implementierung berücksichtigt wurden genauer erläutert. Die Software wurde in Python geschrieben und analysiert bislang Python-Exploits. Eine detailliertere Begründung zur Wahl der Programmiersprache findet sich im Abschnitt 6.1.7.

5.1 Softwaredesign

5.1.1 Architektur

Abbildung 5.1 illustriert die allgemeine Architektur der ExploitDB-Analysesoftware. Ferner wird der Datenfluss durch die Pfeile symbolisiert. Während das GitHub Repository von ExploitDB und MISP externe Komponenten darstellen, sind alle weiteren Komponenten Bestandteil der Software. Zur Anbindung an die externen Komponenten gibt es jeweils Schnittstellen die im folgenden Unterkapitel genauer beschrieben werden.

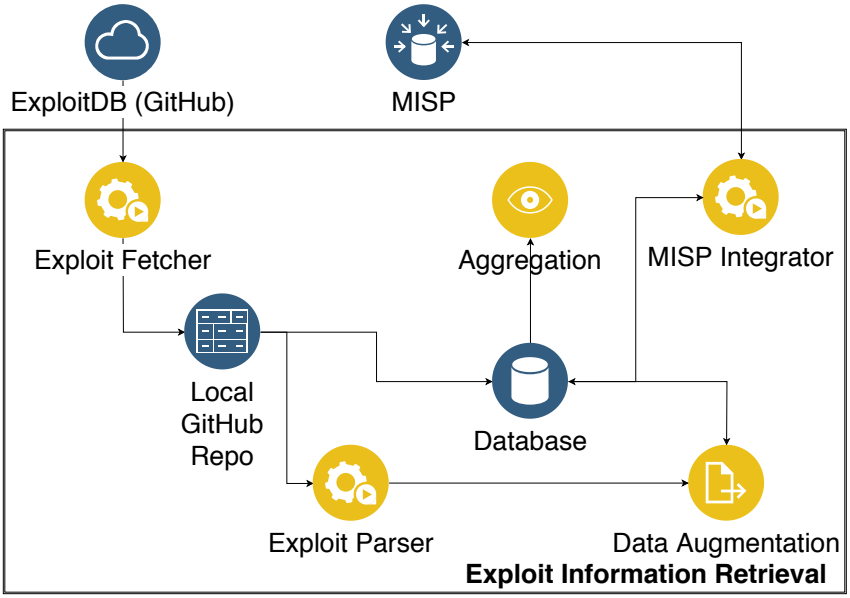

Abbildung 5.1
Architektur der ExploitDB-Analysesoftware

5.1.2 Komponentendiagramm

In Abbildung 5.2 ist das Komponentendiagramm der ExploitDB-Analysesoftware in UML-Notation dargestellt. Die nachfolgenden Abschnitte gehen auf die einzelnen Komponenten detaillierter ein. Sie erläutern zudem den Zusammenhang mit den funktionalen sowie einigen nicht-funktionalen Anforderungen.

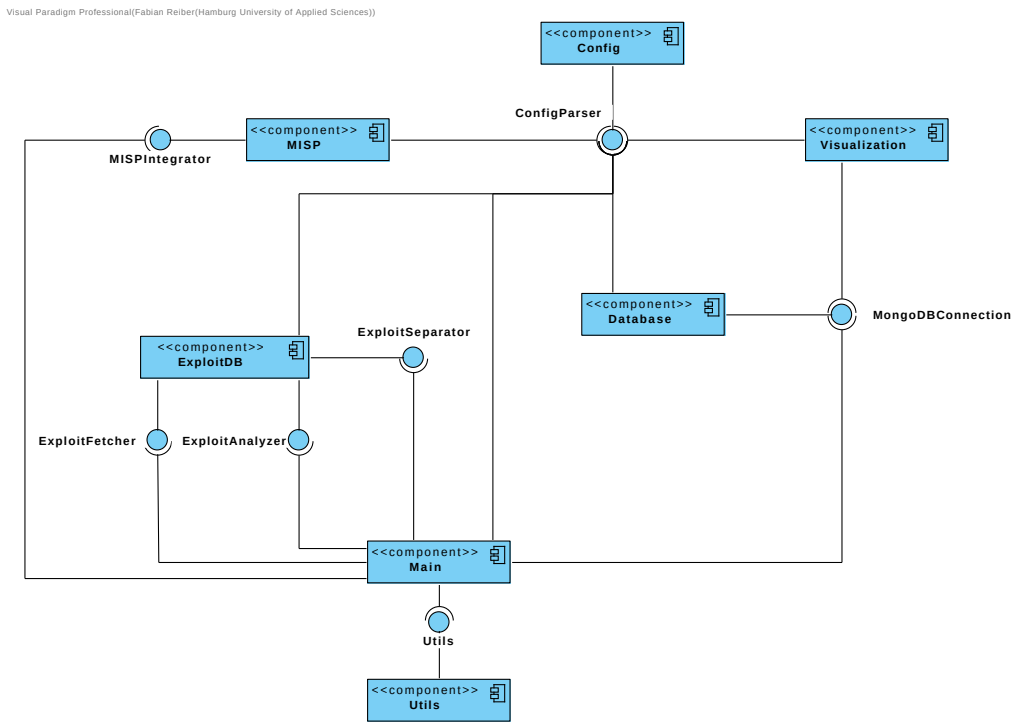

Abbildung 5.2
Komponentendiagramm der ExploitDB-Analysesoftware

Anhand des Komponentendiagramms ist ersichtlich, dass die Architektur sehr modular aufgebaut ist. Weitere Komponenten können problemlos ergänzt oder ausgetauscht werden. Daher erfüllt es die nicht-funktionale Anforderung NFA01 (Änderbarkeit). Beispielsweise kann die Datenbank ausgetauscht werden. Durch den Erhalt der Schnittstellen, erfahren die anderen Komponenten nicht, welche Art von Datenbank in der Komponente zum Einsatz kommt. Eine weitere Änderung kann die Ergänzung einer zusätzlichen Komponente zur Integration der Ergebnisse in eine andere Plattform als MISP sein. Neben der Ergänzung dieser Komponente, muss diese lediglich in der Main-Komponente aufgerufen werden. Ferner bestehen alle Komponenten aus mindestens einem Modul. Darin sind die Funktionen zur Umsetzung ihrer jeweiligen Aufgaben in der Regel durch mindestens eine Klassendefinition realisiert.

Main

Mithilfe der *Main*-Komponente wird der gesamte Ablauf der Analyse gesteuert. Daher nutzt sie jede der anderen bereitgestellten Schnittstellen. Die *Main*-Komponente besteht aus einem Modul in dem sequentiell alle notwendigen Funktionen zur Analyse der Exploits aufgerufen werden. Das Modul kann in fünf Abschnitte eingeteilt werden.

Der erste Abschnitt befasst sich mit der korrekten Initialisierung und Instanziierung aller notwendigen Objekte für die Analyse. Zunächst wird der *ConfigParser* initialisiert, da hierüber wichtige Werte für die Instanziierung weiterer Komponenten bezogen werden. Beispielsweise gibt es einen Abschnitt zur Konfiguration des Loggings. Nahezu jede Komponente setzt ein eigenes Logging ein, welches in der *Main*-Komponente instanziiert und in die jeweiligen Komponenten injiziert wird. Anschließend wird ein Datenbankobjekt für die Verbindung zur Datenbank instanziiert. Durch die Schnittstelle *MongoDBConnection*, ist bereits ersichtlich, dass eine MongoDB eingesetzt wird. Dazu in Kapitel 5.2.2 mehr. Darauf folgend werden die Objekte zum Separieren und der Kommentar- sowie Codeanalyse durch die Schnittstellen *ExploitSeparator* bzw. *ExploitAnalyzer* generiert. Danach wird durch die Schnittstelle *MISPIntegrator* ein Objekt zur Verbindung mit einer MISP-Instanz erzeugt. Schlägt eine der Initialisierungen fehl, bricht die Analyse ab.

Im zweiten Abschnitt werden die Exploits mithilfe der Schnittstelle *ExploitFetcher* aus dem GitHub Repository heruntergeladen. Wird das Repository zum ersten Mal lokal erzeugt, werden im Anschluss alle Exploits aus der Datenbank geladen. Existiert das Repository bereits lokal, werden ausschließlich die neuen Exploits aus der Datenbank bezogen.

Liegt eine Menge von Exploits vor, wird im dritten Abschnitt zunächst die Separierung von Kommentaren und vom Quellcode der Exploits angestoßen. Im Anschluss wird die Kommentar- sowie Codeanalyse durchgeführt. Während der Separierung und Analyse können Fehler auftreten, sodass eine Fehlerbehandlung an dieser Stelle notwendig ist. Sobald ein Fehler auftritt, wird dieser mit der Angabe des betroffenen Exploits geloggt. Außerdem wird der jeweilige Fehler gezählt, sodass am Ende der Analyse ersichtlich ist, wie viele Exploits nicht separiert bzw. analysiert werden konnten.

Während des vierten Abschnittes greift die *Main*-Komponente auf die *Utils*-Schnittstelle zu. Die Kommentar- und Codeanalyse erzeugen jeweils ein Python-Dictionary mit zum

Teil identischen Schlüsseln. Die Werte zu einem identischen Schlüssel müssen für die Integration in MISP zusammengefasst werden. Zudem werden doppelt ermittelte Werte für unterschiedliche Schlüssel, in dem zusammengefassten Dictionary zu einem Schlüssel-Wert-Paar zusammengefasst. Ein weiterer Schritt in diesem Abschnitt stellt die Erzeugung von zusätzlichen Statistiken zu der Kommentar- und Code-Anaylse dar. Ausführlichere Beschreibungen dieser Schritte lassen sich aus den folgenden Abschnitten entnehmen.

Im fünften Abschnitt werden die Exploits erneut aus der Datenbank bezogen und durch die Schnittstelle *MISPIntegration* zu der konfigurierten MISP-Instanz exportiert. Der erneute Abruf der Exploits ist dadurch bedingt, dass diese während der Analyse mit den ermittelten Ergebnissen angereichert wurden. Daher sind die zu Beginn der Analyse ermittelten Exploits nicht mehr aktuell.

Config

Die *Config*-Komponente besteht aus einem Modul und stellt ein Parserobjekt für die allgemeine Konfiguration bereit. Dieses wird nahezu von allen weiteren Komponenten genutzt. Ebenfalls stellt es ein Objekt zum Auslesen einer spezifischen MISP-Konfiguration zur Verfügung. Innerhalb dieser Konfiguration werden die MISP-Attribute zu einem Event konfiguriert. Eine genauere Übersicht und Beschreibung zu den jeweiligen Konfigurationen zeigt Abschnitt 5.2.3 auf.

Database

Durch die *Database*-Komponente wird eine Schnittstelle zwischen der Datenbank und der Analysesoftware bereitgestellt. Sie besteht aus einem Modul und wird durch die *Main*- und *Visualization*-Komponente genutzt. Das Modul greift auf die Konfiguration zu, damit es notwendige Parameter zur Herstellung einer Datenbankverbindung beziehen kann. Es besteht u.a. die Möglichkeit, Exploits in die Datenbank einzupflegen oder bestehende zu aktualisieren. Außerdem können Exploits gefiltert nach einer bestimmten Dateiendung aus der Datenbank bezogen werden. Dadurch können explizit z.B. Python-Exploits aus der Datenbank abgefragt werden.

ExploitDB

Die *ExploitDB*-Komponente enthält drei Module die jeweils die Schnittstellen *ExploitFetcher*, *ExploitSeparator* sowie *ExploitAnalyzer* anbieten. Diese werden ausschließlich von der *Main*-Komponente genutzt. Die folgenden Absätze stellen die jeweiligen Module und ihre Zuständigkeiten genauer vor.

Das *ExploitFetcher*-Modul erfüllt zum einen die funktionale Anforderung FA01. Sie lädt die Exploits aus dem GitHub Repository herunter und speichert diese lokal ab. Zum anderen realisiert es die Anforderung FA02. Es parst die vorliegende CSV-Datei für bereits existierenden Metadaten und speichert diese in der Datenbank ab. Dafür bekommt es ein Datenbankobjekt injiziert, sodass es keine Kopplung zu der *Database*-Komponente aufweist. Weiterhin muss innerhalb des Moduls unterschieden werden, ob bereits eine lokale Kopie des Repositories existiert oder nicht. Sofern das lokale Repository nicht existiert wird die gesamte CSV-Datei ausgelesen und abgespeichert. Existiert das Repository bereits, werden die neuen bzw. modifizierten Metadaten zu einem Exploit ermittelt.

Die Separierung von Kommentaren und Code innerhalb der Exploitdateien wird durch das *ExploitSeparator*-Modul realisiert und erfüllt die Anforderung FA04. Dies ist für die weitere Analyse notwendig, da die Kommentaranalyse mithilfe von String- sowie Pattern-Matching durchgeführt wird. Die Codeanalyse hingegen erzeugt zu jedem Exploit einen abstrakten Syntaxbaum und analysiert diesen. Das Modul weist eine nach dem Factory-Pattern implementierte Basisklasse auf. Dies ermöglicht die Erzeugung von Objekten zur Separierung in Abhängigkeit der Programmiersprache in der die zu separierenden Exploits verfasst sind. Die Basisklasse besitzt zudem die abstrakte Methode *separate(filename)*, sodass jeder sprachspezifische Separator eine eigene Routine zum Separieren implementieren kann. Zur Zeit werden ausschließlich Python-Exploits analysiert, sodass das Modul die Klasse *PythonSeparator* enthält. Nachdem eine Unterscheidung zwischen Kommentaren und Code eines Exploit vorgenommen wurde, werden diese in eine Datei in unterschiedliche Ordner abgespeichert. Dadurch kann die jeweilige Analyse auf den gesonderten Dateien durchgeführt werden. Der Pfad zur Abspeicherung der Dateien wird über die *ConfigParser*-Schnittstelle ermittelt. Sofern ein Exploit nicht erfolgreich separiert werden konnte, wird dieser gesondert in einem Quarantäneordner abgelegt. Dadurch kann manuell geprüft werden, woran der Vorgang gescheitert ist.

Das *ExploitAnalyzer*-Modul enthält die Basisklasse *ExploitAnalyzer*. Diese wird von den Klassen *CommentAnalyzer* sowie *CodeAnalyzer* geerbt, welche die Anforderungen FA05

bzw. FA06 erfüllen. Die Basisklasse besitzt zudem die abstrakte Methode *analyze(exploit, filename, exploit_ id)*, sodass jede Klasse eine individuelle Analyse durchführen muss. Zudem wird innerhalb dieser Methode eine allgemeine Analyse realisiert. Bei der manuellen Sichtung hat sich gezeigt, dass diverse Pattern sowohl in den Kommentaren als auch im Code gleichermaßen auftreten. Bislang wird ein Pattern-Matching auf CVE, IP-Adressen, URLs sowie E-Mail Adressen angewendet. Eine weitere, von der Basisklasse, bereitgestellte Funktion ist *generate_ stats()*. Wie bereits erwähnt werden diese Statistiken nach der Analyse auf der Konsole ausgegeben. Sie dienen dazu eine Übersicht zu den ermittelten Ergebnisse zu erhalten. Somit ist am Ende der Analyse bspw. ersichtlich wie viele CVEs oder IP-Adressen nach einem Durchlauf der Software ermittelt werden konnten. Das Modul nutzt außerdem die *ConfigParser*-Schnittstelle u.a. zur Ermittlung der Ordner in denen sich die vom *ExploitSeparator* erzeugten Dateien befinden. Weiterhin werden die zu analysierenden Informationen aus der Konfiguration bezogen. Dadurch, dass die Kommentaranalyse ein String-Matching realisiert, werden innerhalb der Konfiguration die in den Kommentaren zu suchenden Hinweise angegeben, die bei der manuellen Durchsicht ermittelt wurden. Zum Beispiel kann durch den Hinweis *email* bzw. *e-mail* eine in den Kommentaren angegebene E-Mail Adresse extrahiert werden. Eine ausführlichere Darstellung dieser Hinweise wird im Abschnitt 5.2.3 gegeben.

MISP

Zur Umsetzung von FA07 wird innerhalb der MISP-Komponente eine Verbindung zu einer MISP-Instanz aufgebaut. Die Komponente besteht aus einem Modul und bietet eine Schnittstelle an, welche von der *Main*-Komponente genutzt wird. Zur Entkopplung bekommt sie, wie das *ExploitFetcher*-Modul, ein Datenbankobjekt injiziert. Die Datenbank bietet eine Funktion an, sodass jedem Exploit ein neues Schlüssel-Wert-Paar zugewiesen werden kann. Während der Integration wird jeder Exploit mit einem neuen Paar versehen aus dem ersichtlich ist, ob ein Exploit bereits nach MISP exportiert wurde. Außerdem greift das Modul auf die spezifische MISP-Konfiguration zu. Diese dient zur Generierung von Attributen zu den jeweiligen Events. Eine genauere Erläuterung zu der Konfiguration ist in Abschnitt 5.2.3 angegeben. Das Modul besitzt die Klasse *MISPIntegrator* in der die Methode *integrate(exploits)* definiert ist. Sie erzeugt aus jedem der übergebenen Exploits ein neues Event und fügt ihnen die analysierten Ergebnisse als MISP-Attribut hinzu. Weiterhin wird jedem Event ein für diese Software spezifischer Tag zugewiesen, welcher in der allgemeinen Konfiguration hinterlegt ist. MISP bietet, wie in Kapitel 2.4

erwähnt, die Möglichkeit jedes Event mithilfe von Taxonomien zu Klassifizieren. Diese werden ebenfalls in der Konfiguration definiert und zu jedem Event hinzugefügt. Eine Beschreibung der einegsetzten Taxonomien ist in Abschnitt 5.2.7 aufgeführt.

Visualization

Die *Visualization*-Komponente erfüllt die Anforderung FA03 und besteht aus zwei Modulen. Sie sind von der Analyse der Exploits losgelöst und werden separat gestartet. Aus diesem Grund nutzt sie die *MongoDBConnection*-Schnittstelle und bezieht die Exploits aus der Datenbank. Unter anderem wird mithilfe dieses Moduls ein Diagramm erzeugt, welches die Jahre mit den meisten Exploits gruppiert nach der genutzten Programmiersprache darstellt. Die erzeugten Diagramme werden als Bilddatei in einem Verzeichnis, welches durch die Konfiguration angegeben wird, abgespeichert. Durch ein zweites Modul werden diverse Statistiken aus den Daten in der Mongo Datenbank erstellt, wie z.B. die Summe der einzelnen Hinweise für alle Python-Exploits in einem angegebenen Zeitraum.

Utils

Für die *Main*-Komponente gibt es die Hilfskomponente *Utils*. In ihr werden, wie schon erwähnt, einige Funktionalitäten für die Analyse bereitgestellt. Sie haben nicht direkt etwas mit der Analyse zu tun und werden daher unabhängig in eine zusätzliche Komponente ausgelagert.

5.1.3 Datenmodell

Abbildung 5.3 zeigt das Datenmodell auf. Es stellt zum einen Objekte dar und zum anderen welche Daten diese produzieren bzw. erhalten. Das Objekt *ExploitDB Fetcher* lädt die Exploits von GitHub herunter und extrahiert Metadaten aus der vorliegenden CSV-Datei. Das erzeugte Datenobjekt in der Abbildung stellt die Metadaten zu jedem Exploit als JSON-Objekt dar. Es gibt zudem die Datentypen für jeden Wert des jeweiligen Schlüssels an. Dieses Datenobjekt wird sodann an das *Database*-Objekt übergeben und in der Datenbank abgespeichert. Das *Comment Analyzer*-Objekt reichert jedes Datenobjekt mit weiteren Schlüssel-Wert-Paaren an. Daher werden in dem Datenobjekt die bereits vorhandenen Metadaten mit abgebildet. In Abschnitt 5.2.3 wird genauer auf die

weiteren Schlüssel-Wert-Paare eingegangen. Zuletzt reichert das *Code Analyzer*-Objekt das Datenobjekt mit weiteren Informationen an. Da während der Codeanalyse z.B. nach Variablen gesucht wird die das Wort *port* enthalten, wird die gesamte Bezeichnung als Schlüssel in das Datenobjekt hinzugefügt. Daher ist es nicht möglich diese Schlüssel-Wert-Paare in der Abbildung festzuhalten.

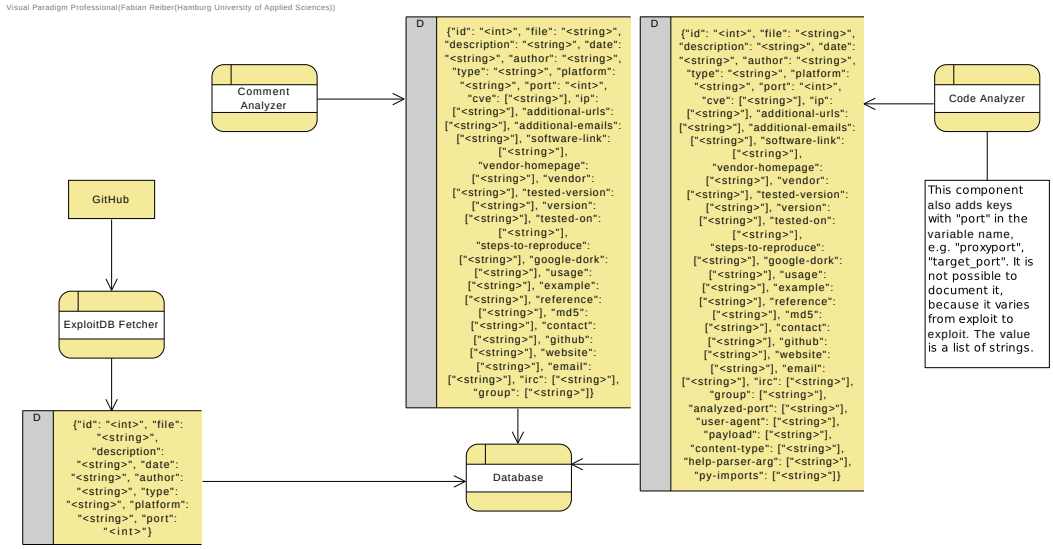

Abbildung 5.3
Datenmodell der ExploitDB-Analysesoftware

5.2 Technische Realisierung

In diesem Kapitel werden spezifischere Details zu der Implementierung der ExploitDB-Analysesoftware ausgeführt. Die Software wurde auf Debian GNU/Linux 9 (stretch) entwickelt, ausgeführt und getestet. Sie ist in Python 2.7.13 entwickelt worden. Python wurde u.a. gewählt, da MISP eine in Python verfasste REST-Schnittstelle anbietet, welche sodann problemlos einzubinden ist. Die Analysesoftware ist öffentlich zugänglich und

liegt im GitHub Repository des Autors dieser Arbeit [78]. In der dortigen *README.md* sind die Details zur Installation sowie Inbetriebnahme angegeben.

5.2.1 Eingesetzte Programmierbibliotheken

Einige für die Realisierung verwendeten Python-Bibliotheken werden im Folgenden kurz beschrieben.

GitPython

GitPython bietet eine Schnittstelle in Python für den Zugriff auf GitHub Repositories an [94]. Sie wird in der Version 2.1.1 eingesetzt. Mit dieser Bibliothek lassen sich die notwendigen Operationen u.a. für das Klonen eines Repositories oder der Ermittlung von Unterschieden einzelner Dateien zwischen verschiedenen Commits nutzen. Letztere Funktionalität ist notwendig damit neue oder aktualisierte Metadaten zu einem Exploit aus der CSV-Datei extrahiert werden können.

PyMongo

PyMongo in der Version 3.8.0 bietet der Analysesoftware den Zugriff auf die MongoDB an [51]. Mit dieser Bibliothek wird über ein sogenanntes MongoClient-Objekt auf die angegebene Datenbank zugegriffen. Damit lässt sich sodann auf eine Collection zugreifen. Eine Collection umfasst eine Menge von Dokumenten in denen jeweils die Informationen zu einem Exploit abgespeichert werden.

Parso

Zur Einteilung der Exploits in Kommentaren und Code wurde die Bibliothek *parso* in Version 0.4.0 eingesetzt [24]. Mit ihr ist es möglich den Quellcode zu parsen, sodass dieser als Baumstruktur vorliegt. Mit der Bibliothek ist es zudem möglich den Quellcode für unterschiedliche Pythonversionen zu parsen, da die Exploits sowohl in Python 2 als auch in Python 3 verfasst sind. Sobald der Code eingelesen und die Baumstruktur erzeugt wurde, kann dieser rekursiv analysiert und zwischen den Kommentaren und dem Code unterschieden werden.

PyMISP

Die Integration neuer Events in MISP erfolgt über die bereitgestellte REST-Schnittstelle. Diese wird durch die Bibliothek *pymisp* in Version 2.4.106 genutzt [95]. Dadurch können bspw. Events erzeugt und ihnen neue Attribute zugewiesen werden.

Pandas/Matplotlib

Zur Erzeugung und Visualisierung der Statistiken wurde *pandas* in Version 0.23.4 und *matplotlib* in Version 2.2.4 eingesetzt [32, 25]. *Pandas* ist eine Bibliothek zur einfachen und effizienten Analyse von Daten. Somit können die Informationen zu den Exploits bspw. nach dem Datum sortiert und nach einem bestimmten Merkmal gefiltert werden. *Matplotlib* erzeugt hoch qualitative Bilder aus Objekten die durch *pandas* erzeugt wurden. Mithilfe dieser Bibliothek lassen sich die Bilder auf der Festplatte speichern.

5.2.2 Datenbankanbindung

Wie zuvor erwähnt wird eine Mongo Datenbank für die Abspeicherung der analysierten Ergebnisse eingesetzt [50]. Hierbei handelt es sich um eine NoSQL-Datenbank, welche, im Gegensatz zu SQL-Datenbanken, kein vordefiniertes Datenschema benötigt. Dies ist notwendig, da bei der Analyse der Exploits nicht immer die identischen Schlüssel und die gleiche Anzahl von Schlüssel-Wert-Paaren ermittelt werden. Die Daten werden in der Datenbank als JSON-ähnliche Objekte abgespeichert. Dies vereinfacht z.B. das Einfügen neuer Informationen von Exploits. Die Informationen zu jedem Exploit werden in einem Python-Dictionary festgehalten und in einer Liste abgelegt. Diese Liste kann sodann an die Datenbank übertragen werden und bedarf keiner weiteren Konvertierung. Somit ist die nicht-funktionale Anforderung NFA04 (Fehlertoleranz) erfüllt.

5.2.3 Konfiguration der Analysesoftware

Insgesamt gibt es vier Konfigurationsdateien für die Analysesoftware. Die allgemeine Konfiguration besteht aus den Abschnitten *analyzer*, *git*, *logging*, *misp*, *mongodb*, *separator* sowie *visualization* und stellt diverse Parameter für die jeweiligen Komponenten der Software bereit.

Die Komponente zur Analyse der Exploits greift u.a. auf die in der Konfiguration definierten Hinweise zu. Diese werden zum Beispiel zur Kommentaranalyse für ein String-Matching genutzt. Gleichzeitig dient ein Hinweis zur Abbildung auf die Kategorie (*category*) und den Typen (*type*) eines MISP-Attributes. Wie bereits im MISP Core Format erläutert, sind *category* und *type* zwei von drei Schlüssel-Wert-Paaren die ein Attribut zu einem Event definieren. Das dritte Schlüssel-Wert-Paar Wert (*value*) ist der ermittelte Wert aus der Analyse. Ausschnitt 5.1 zeigt die MISP-spezifische Konfigurationsdatei in der jedem Hinweis eine Kategorie und einen Typen zugewiesen wird. Die Werte zu den Hinweisen *id*, *type*, *platform* und *port* stammen aus der CSV-Datei. Den Hinweisen *file*, *description*, *date* und *author* aus der CSV-Datei werden keine Schlüssel-Wert-Paare zugewiesen, da sie zum Anlegen von Events eine andere Funktion erfüllen. Dies wird in Abschnitt 5.2.7 erläutert. *Analyzed-port*, *user-agent*, *payload*, *content-type*, *help-parser-arg* und *py-imports* sind jeweils Hinweise zu den Werten aus der Codeanalyse. Wie diese ermittelt werden, wird in Abschnitt 5.2.6 genauer beschrieben. Alle weiteren Hinweise werden während der Kommentaranalyse eingesetzt. In 5.2.5 wird nochmals genauer darauf eingegangen. Abbildung 5.4 zeigt exemplarisch die Liste von ermittelten MISP-Attributen zu dem Exploit „MP4 Converter 3.25.22 - 'Name' Denial of Service (PoC)“ [66].

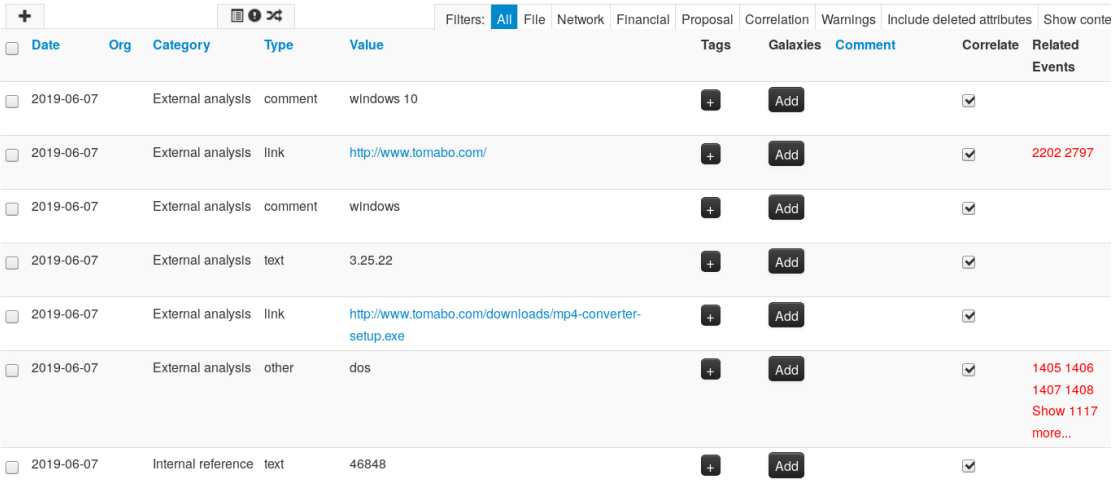

Abbildung 5.4
Beispielhaft die erzeugten MISP-Attributes eines Exploits

```
[attributes]
id = Internal reference,text
type = External analysis,other
platform = External analysis,comment
port = External analysis,port
cve = External analysis,vulnerability
ip = Network activity,ip-dst
additional-urls = External analysis,link
additional-emails = Social network,email-src
software-link = External analysis,link
vendor-homepage = External analysis,link
vendor = External analysis,other
tested-version = External analysis,text
version = External analysis,text
tested-on = External analysis,comment
stept-to-reproduce = External analysis,comment
google-dork = External analysis,other
usage = External analysis,other
example = External analysis,other
reference = External analysis,text
md5 = External analysis,md5
contact = Social network,text
github = Social network,github-repository
website = External analysis,link
email = Social network,email-src
irc = Social network,text
group = Attribution,threat-actor
analyzed-port = External analysis,port
user-agent = External analysis,user-agent
payload = External analysis,other
content-type = Payload delivery,mime-type
help-parser-arg = External analysis,comment
py-imports = External analysis,other
```

Codeausschnitt 5.1
Zuweisung von Hinweisen zu Kategorie und Typ eines MISP-Attributes

5.2.4 Trennung von Kommentaren und Code

Die Notwendigkeit eine Separierung von Kommentaren und Code der Exploits durchzuführen ergibt sich daraus, dass die Erzeugung von Abstract Syntax Trees (ASTs) mithilfe der Bibliothek *ast* in Python, zu konservativ ist. Das heißt, dass bei diversen Syntaxfehlern innerhalb eines Moduls, kein AST mehr erzeugt werden kann. Gleiches gilt für nicht-validen Pythoncode innerhalb eines Exploits. Es hat sich zu Beginn der Entwicklung gezeigt, dass sehr viele Exploits davon betroffen waren und es somit zu deutlich geringeren Ergebnissen führen würde. Ein weiterer Grund ist, dass *ast* nur ASTs aus Modulen erzeugen kann, die in der selben Hauptversion, wie die des auszuführenden Interpreters, vorliegen.

Dies sind bereits zwei Anforderungen die von der Bibliothek *parso* erfüllt werden. Zu Beginn des Spearierungsprozesses wird geprüft in welcher Pythonversion der Exploit verfasst ist. Dies geschieht durch die Prüfung, ob eine Angabe zu einem Python3-Interpreter in dem Exploit vorhanden ist. Wenn ja wird dem *parso*-Konstruktor eine Python3-Version übergeben. Andernfalls wird er ohne jegliche Versionsangabe aufgerufen und kann somit Pythoncode in Version 2 parsen.

Durch das Parsen eines Exploit-Moduls, entsteht ein *parso*-Baum, über den rekursiv iteriert wird. Dies geschieht so lange bis ein Knoten erreicht wurde der ein Blatt des Baumes repräsentiert. Ein dedizierter Kommentarknoten stellt *parso* allerdings nicht zur Verfügung. Daher muss an dieser Stelle separat zwischen dem Kommentar und dem Code unterschieden werden. Um dies zu gewährleisten bietet *parso* die Möglichkeit an, sich zu jedem Knoten den Code zurück geben zu lassen. Dieser kann mit und ohne Kommentar ermittelt werden. Daher ist es notwendig, den Kommentar daraus zu extrahieren. Ferner werden die Zeilen mit nicht-validen Pythoncode als Kommentare behandelt. Verbliebe dieser Code im Exploit, könnte mit der Bibliothek *ast* kein abstrakter Syntaxbaum erzeugt werden.

Während der Bearbeitung der Baumknoten können jederzeit Syntaxfehler von *parso* erkannt werden. Trifft dies zu, wird dies geloggt und der bereits separierte Exploit in einen gesonderten Quarantäneordner gespeichert. Dort kann er manuell betrachtet und ggf. korrigiert werden.

5.2.5 Kommentaranalyse

In Abschnitt 2.4 wurde bereits erwähnt, dass die Exploits mit diversen Metadaten im Kommentar versehen sein sollen. Codeausschnitt 5.2 zeigt jene von Offensive Security gewünschten Metadaten die mindestens angegeben werden sollten.

```
# Exploit Title: [title]
# Google Dork: [if applicable]
# Date: [date]
# Exploit Author: [author]
# Vendor Homepage: [link]
# Software Link: [download link if available]
# Version: [app version] (REQUIRED)
# Tested on: [relevant os]
# CVE : [if applicable]
```

Codeausschnitt 5.2
Gewünschte Metadaten zu jedem Exploit [64]

Innerhalb der Kommentaranalyse wird nach diesen und weiteren Hinweisen durch ein String-Matching gesucht. Mit Ausnahme von *Exploit Title*, *Date* und *Exploit Author*, da diese schon in der CSV-Datei angegeben sind. Alle weiteren Hinweise wurden durch eine manuelle Sichtung verschiedener Exploits ermittelt. Das Finden der Hinweise wird mithilfe der Bibliothek *re* durchgeführt mit der reguläre Ausdrücke auf Strings angewendet werden können [76]. *Re* stellt die Methode *finditer(pattern, string, flags=0)* zur Verfügung mit der alle in einem Exploit gefundenen Pattern in einer Liste zusammengefasst zurück gegeben werden. Die jeweiligen Ergebnisse zu einem Hinweis werden anschließend weiter untersucht und ggf. überflüssige Zeichen entfernt. Es kann vorkommen, dass Ergebnisse bspw. eine Raute am Anfang des Strings enthalten. Rauten werden in Python als einzeilige Kommentare eingesetzt. Daher wurde ein Hinweis gefunden, das Ergebnis hingegen ist in der darunterliegenden Zeile zu finden. Diese Vereinheitlichung der Ergebnisse können etwaige spätere Analysen begünstigen.

Eine weitere Besonderheit, ist die Angabe einer Gruppenbezeichnung innerhalb eines Kommentars. Bislang wurde in den manuellen Analysen nur eine Gruppe ausgemacht, die regelmäßig Exploits veröffentlicht. Ein Mechanismus zum Auffinden dieser Gruppen wurde ebenfalls in die Kommentaranalyse integriert. In der MISP-spezifischen Konfiguration werden die Ergebnisse zu einem Exploit mit dem Hinweis *group* zusammengefasst.

5.2.6 Codeanalyse

Die Codeanalyse nutzt die Bibliothek *ast* zur Bildung eines abstrakten Syntaxbaumes [75]. Die Bibliothek wurde gewählt, da die Knoten des ASTs automatisch kategorisiert werden. Dies bedeutet z.B., dass eine Import-Anweisung durch die Klasse *ast.Import* repräsentiert wird. Oder die Definition eines Python-Dictionaries als *ast.Dict*. Dies vereinfacht die Analyse eines Exploits in dem Sinne, dass der Quellcode einfacher zu extrahieren und zu durchsuchen ist. Gerade im Hinblick auf Code der sich über mehrere Zeilen erstreckt. Codeausschnitt 5.3 zeigt beispielhaft den Ablauf zum Extrahieren und Speichern der importiertern Module innerhalb eines Exploits.

```
elif isinstance(node, ast.Import):
    for imp in node.names:
        self._result_add(filename, 'py-imports', imp.name)
```

Codeausschnitt 5.3
Extrahieren importierter Module innerhalb eines Exploits

Die manuelle Durchsicht der Exploits ergab weitere Erkenntnisse zur Analyse des Quellcodes. Beispielsweise wird bei der Analyse von Python-Dictionaries geprüft, ob es Schlüssel gibt die einen *user-agent* oder einen *content-type* enthalten. Die Bibliothek *ast* bietet einen direkten Zugriff auf den Wert eines *ast.Dict*-Knotens unter Angabe des Schlüssels. Somit ist das Extrahieren der Werte, in diesem Fall von *user-agent* und *content-type*, problemlos zu realisieren.

Weitere genutzte Knotentypen sind *ast.ImportFrom*, *ast.Assign*, *ast.Expr* sowie *ast.AugAssign*. Mit *ast.Assign*-Knoten werden Variablenzuweisungen untersucht. Während der Analyse wird ebenfalls nach *user-agent* und *content-type* gesucht. Zudem wird nach Variablenbezeichnungen gesucht die den Begriff *port* oder *payload* enthalten. Als Ergebnis wird die Variablenbezeichnung als Schlüssel und der jeweils ermittelte Port als Wert gespeichert. Eine Payload wird hingegen immer unter dem Schlüssel *payload* abgespeichert.

Ast.AugAssign sind u.a. Zuweisungen der Form *msg += 'something'*. Bislang wird hierfür nach Variablen gesucht die den Bezeichner *content-type* aufweisen. Die Werte für *user-agent* sowie *content-type* werden zudem aus *ast.Expr*-Knoten extrahiert. Des Weiteren wird in einer Expression nach dem Wort *help* als Schlüssel gesucht. Dies hat den Grund, dass einige Exploits einen *ArgumentParser* einsetzen. Dieser gibt zur Ausführung der

Exploits an, welche Argumente diesem übergeben werden müssen. Während der Initialisierung eines *ArgumentParsers* kann bei der Angabe der zu übergebenen Argumente ein Schlüssel-Wert-Paar zur Definition eines Hilfetextes angegeben werden dessen Schlüssel *help* ist. Während der Analyse wird der angegebene Wert - der Hilfetext - extrahiert und als Ergebnis unter dem Schlüssel *help-parser-arg* abgespeichert. *Ast.ImportFrom* repräsentiert den Import einzelner Bestandteile (Module, Klassen, Methoden, etc.) aus einer Bibliothek. Diese werden, wie die bereits erwähnten Importanweisungen, unter den Schlüssel *py-imports* abgespeichert.

Des Weiteren war es notwendig eine eigene Routine zur Codeanalyse zu entwickeln, da vorhandene Codeanalysetools für Python nicht den Anforderungen der Exploit-Analyse gerecht werden. Diese Tools bieten die Möglichkeit den Pythoncode auf Fehler zu überprüfen, die Codekomplexität zu berechnen oder den Quellcode nach Schwachstellen zu überprüfen. Eine kleine Auswahl an Tools bietet Endler [12]. Bei den hier aufgestellten Anforderungen an eine Exploit-Analyse, bedarf es allerdings der konkreten Untersuchung einzelner Bestandteile im Quellcode und weniger, ob diese korrekt sind oder welche Komplexität sie aufweisen.

5.2.7 Integration in MISP

Die durch die Komponente MISPIntegrator erzeugten Events wurden während der Entwicklung in eine lokale MISP-Instanz, die in einer virtuellen Maschine betrieben wurde, integriert. Hierfür wurde MISP in Version 2.4.95 eingesetzt. Über die allgemeine Konfiguration wird die URL, über die die Instanz zu erreichen ist, angegeben.

Wie im Unterkapitel 2.3.1 erwähnt, benötigt jedes Event in MISP die Attribute *info* und *date*. Das in der CSV-Datei enthaltene Attribut *description* wird für das Event-Attribut *info* genutzt. Ein Datum zu jedem Exploit und somit jedem Event ist ebenfalls in der Datei enthalten (*date*).

Sobald die Informationen zu einem Exploit nach MISP exportiert wurden, wird zu einem Exploit in der Datenbank eine Universally Unique Identifier (UUID) hinzugefügt. Dies gewährleistet, dass ein Event zu einem Exploit nicht mehrfach erzeugt und es ggf. aktualisiert wird, sobald der Exploit in der ExploitDB aktualisiert wurde. Zudem werden die in dem Quarantäneordner befindlichen Exploits nach MISP integriert.

Wurde ein neues Event angelegt, wird es zur Unterscheidung in MISP zudem getaggt. Der Tag wird in der allgemeinen Konfiguration angegeben. Weiterhin wurden diverse Taxonomien aus der MISP-Dokumentation ausgewählt, welche jedes neue Event automatisch erhält. Eine Übersicht zu den ausgesuchten Taxonomien zeigt Tabelle 5.1. Die Auswahl der Taxonomien beruht darauf, dass in der jeweiligen Bezeichnung bzw. ausführlicheren Beschreibung der Begriff "Exploit" verwendet wurde. Eine differenzierte Unterscheidung, ob eine Taxonomie zu 100% zu einem Exploit passt, wird derzeit nicht vorgenommen.

Bezeichnung	**Kurze Beschreibung**
CERT-XLM:intrusion-attempts="exploit-known-vuln"	Exploiting known vulnerabilities [44]
CERT-XLM:intrusion-attempts="new-attack-signature"	New attack signature [45]
cyber-threat-framework:Engagement="exploit-vulnerabilities"	Exploit vulnerabilities [46]
enisa:nefarious-activity-abuse="exploits-exploit-kits"	Exploits/Exploit Kits [47]
Europol-event:exploit-tool-exhausting-resources	Exploit or tool aimed at exhausting resources (network, processing capacity, sessions, etc...) [48]

Tabelle 5.1
Eingesetzte MISP Taxonomien

Weiterhin bietet MISP sogenannte Objects zur Kombination von vordefinierten Attributen an. Sie dienen zur genaueren Beschreibung von Events [40]. Zu jedem neuen Event wird daher ein Exploit-POC-Object angefügt [41]. Dieses enthält die Beschreibung des Exploits, den Namen des Autoren oder der Autorin, eine Referenz auf den Exploit in dem GitHub Repository und den Code des Exploits. Die Beschreibung (*description*) und der Name (*author*) stammen aus der verfügbaren CSV-Datei. Die Referenz wird aus dem Link zum GitHub Repository und dem in der CSV-Datei verfügbaren Dateinamen (*file*) gebildet.

5.2.8 Validierung der Funktionalität

Zur Erfüllung von NFA03 (Korrektheit) sowie NFA05 (Prüfbarkeit) wurden für die Komponenten *ExploitDB*, *Database*, *MISP* sowie *Utils* Tests erstellt. Es ist wichtig korrekte Analyseergebnisse zu erhalten, da die Gefahr besteht, dass *false positives* erzeugt werden. Diese können bspw. zu falschen Entscheidungen bei der Erstellung von Sicherheitsstrategien in Unternehmen führen.

Zur Prüfung der Datenbank-Schnittstelle, wird eine Test-Datenbank und Test-Collection erzeugt. Sie werden nach Durchführung der Tests wieder gelöscht. Die Tests prüfen die angebotenen Funktionen des Datenbank-Moduls und ihre in der Datenbank angelegten oder geänderten Ergebnisse.

Die Separierung sowie die Kommentar- und Codeanalyse werden auf manuell erstellte Python-Module angewendet. Bei der Separierung kann es z.B. vorkommen, dass der Quellcode eines Exploits nicht korrekt eingerückt ist. Dies muss währenddessen abgefangen und behandelt werden. Aus diesem Grund wurde dafür ein Testfall erzeugt. Weiterhin muss zum Beispiel zwischen mehrzeiligen Kommentaren und mehrzeiligen Variablenzuweisungen während der Separierung unterschieden werden. Dies ist ebenfalls durch einen Testfall abgedeckt.

Der Test zur Pürfung des ExploitFetcher-Moduls greift nicht auf das existierende GitHub Repository der ExploitDB zu. Es nutzt stattdessen die Bibliothek *mock* zur dessen Simulation [23]. Dadurch können den Funktionen der genutzten Schnittstellen von GitPython individuelle Rückgabewerte zugewiesen werden, die der tatsächlichen entsprechen.

Die Tests zur Prüfung des MISPIntegrator-Moduls greifen ebenfalls auf *mock* zurück, wodurch eine MISP-Instanz simuliert wird. Zudem wird die spezifische Konfigurationsdatei zur Erzeugung von MISP-Attributen getestet. Dadurch ist gewährleistet, dass die korrekten Werte von *category* und *type* eines MISP-Attributes konfiguriert sind. Andernfalls könnte bspw. der Typ von *ip-dst* auf *ip-src* umgestellt werden, welches zu einer falschen Interpretation des Attributes in MISP führen würde.

5.2.9 Zeitverhalten der Datenverarbeitung

NFA02 stellt die Anforderung auf, dass das System in allen Bereichen der Datenverarbeitung effizient und zügig arbeiten soll, da die Informationen möglichst in Echtzeit

bereitgestellt werden sollen. Eine Bearbeitung in Echtzeit ist sehr sinnvoll, damit eine schnelle Reaktion auf etwaige Vorfälle realisierbar ist. Dieser Aspekt wurde bei der Entwicklung nicht weiter in den Fokus gestellt, da nicht festgelegt ist zu welcher täglichen Uhrzeit neue Exploits in der ExploitDB veröffentlicht oder aktualisiert werden. Daher sollte die Analysesoftware im produktiven Betrieb mehrmals täglich ausgeführt werden. Es hat sich bei der ersten Ausführung der Analysesoftware jedoch gezeigt, dass die Integration der Ergebnisse von ca. 2500 Exploits in MISP über eine Stunde gedauert hat. Eine Analyse der Gründe wurde nicht durchgeführt. Letztlich ist es eine einmalig auftretende Zeitspanne, die den Aspekt der Echtzeitverarbeitung nicht berührt, da die Exploits bereits seit einiger Zeit vorhanden sind. Werden dagegen einzelne neue Exploits oder Aktualisierungen verarbeitet, benötigt der Prozess nur wenige Sekunden, bis die Ergebnisse in MISP vorzufinden sind.

6 Evaluation

Alle in diesem Kapitel vorgestellten Statistiken betrachten den Zeitraum zwischen den Jahren 2009 und 2018. Da das Jahr 2019 während der Evaluation noch nicht abgeschlossen war und sich die Statistiken verzerren würden, wurde das Jahr 2018 gewählt. Da in den vergangenen 10 Jahren, von Ende 2018 aus betrachtet, die Entwicklung von Python-Exploits stärker angestiegen ist als in den 21 Jahren davor, wird der Zeitraum dieser letzten 10 Jahre betrachtet. Die Statistiken greifen auf den Datenbestand der ExploitDB vom 04.10.2019 zurück.

6.1 Evaluation der Forschungsfragen

6.1.1 Identifikation von Hinweisen (FF01)

Frage: Welche Hinweise zur Identifizierung von Vorfällen können aus den Exploits der ExploitDB extrahiert werden?

Hypothese: Wenn Exploits Quellen für Hinweise aus den Kategorien *precursor* sowie *indicator* sind, dann können aus den Exploits der ExploitDB ebenfalls Hinweise aus diesen Kategorien extrahiert werden.

Mit der Analyse von Python-Exploits aus der ExploitDB konnten erfolgreich verschiedene Hinweise aus den Kategorien *precursor* und *indicator* ermittelt werden. Somit kann die obige Hypothese bestätigt werden.

Im Zeitraum von 2009 bis 2018 wurden 2406 Python-Exploits gezählt. Wohingegen 282 Python-Exploits zwischen den Jahren 1988 und 2008 veröffentlicht wurden. Diese Werte zeigen einen Anstieg der Entwicklung von Python-Exploits in dieser Dekade, im Vergleich zu den 21 Jahren davor, deutlich auf. Es konnten allerdings nicht alle 2688 Python-Exploits analysiert werden. Wie u.a. das Unterkapitel 6.4 beschreibt, enthalten diverse

Exploits zum Teil nicht validen Pythoncode. Sie wurden in einem separaten Qurantäneordner abgelegt. Für den Zeitraum zwischen 1988 und 2008 konnten 28 Exploits, knapp 9,92%, aus diesem Grund nicht analysiert werden. Während 174 Exploits, knapp 7,23%, im Zeitraum von 2009 bis 2018 als Quarantäne behandelt wurden. Dennoch sind durch die CSV-Datei diverse Metadaten zu diesen Exploits vorhanden. In den Analyseergebnissen in den Tabellen 6.1 und 6.2 auf Seite 82 bzw. 83 werden hingegen nur die Exploits betrachtet, die den gesamten Analyseprozess durchlaufen haben. Somit wird die Anzahl der Exploits in Quarantäne aus diesen Werten herausgerechnet. Dadurch werden die ermittelten Ergebnisse vergleichbar.

Demzufolge werden insgesamt 2486 Python-Exploits betrachtet, wovon 2232 auf den Zeitraum zwischen den Jahren 2009 und 2018 abfallen und 254 auf den Zeitraum zwischen 1988 und 2008.

Tabelle 6.1 zeigt in der ersten Spalte die Bezeichnung der Hinweise auf. Die zweite Spalte stellt die Anzahl der Hinweise im Verhältnis zu der Anzahl der Exploits in denen sie mindestens einmal vorkommen, im Zeitraum zwischen 2009 und 2018, dar. Dabei wurde zusätzlich zwischen dem Zeitraum 1988 - Veröffentlichung des ersten Exploits - und 2008 unterschieden. Dieser ist in der dritten Spalte ausgewiesen. Zusätzlich wurde die Summe beider Zeiträume ermittelt und wird in der vierten Spalte repräsentiert. Zudem illustrieren die Spalten fünf und sechs eine Zuordnung der Hinweise zu den Kategorien *precursor* und *indicator*. Ein Häkchen bedeutet, dass der Hinweis der jeweiligen Kategorie zuzuordnen ist. Ein Kreuz hingegen nicht. So konnten beispielsweise 447 CVE-Nummern aus 373 Exploits, in denen mindestens eine CVE-Nummer vorkommt, zwischen den Jahren 2009 und 2018 ermittelt werden. Insgesamt konnten 461 CVE-Nummern aus 384 Exploits extrahiert werden.

Die Hinweise *id*, *file*, *description*, *date*, *author*, *type* und *platform* kommen in jedem Exploit vor, da diese aus den vorliegenden Metadaten der CSV-Datei stammen. *port* ist ebenfalls Bestandteil dieser Daten, besitzt hingegen nicht für jeden Exploit einen Wert. Alle weiteren Merkmale konnten während der Analyse nicht aus allen 2486 Exploits ermittelt werden.

Die Kategorisierung zeigt, dass jeder Hinweis entweder in die *precursor*- und *indicator*-Kategorie einzuordnen ist, oder gar keiner dieser beiden Kategorien zuzuordnen ist. Die zu jedem Exploit bereits verfügbaren Metadaten *id*, *file*, *description* und *author* können keiner der beiden Kategorien zugeordnet werden. Sie sind als Informationen über einen

Exploit zu sehen und weniger als einen Anhaltspunkt für einen zukünftigen oder aktuellen Vorfall. *date*, *type*, *platform* sowie *port* hingegen können in beide Kategorien zur Identifizierung von Vorfällen einsortiert werden. Aus dem Datum kann bspw. geschlossen werden, ob die Verwendung eines Exploits einen Vorfall verursachte, da dieser zeitlich nach dessen Veröffentlichung beobachtet wurde. Der Typ kann als Indiz eines aktuell durchgeführten DoS-Angriffs aufzeigen, dass jener Exploit dazu genutzt wird oder wurde. Die Plattform weist darauf hin, welches System durch den jeweiligen Exploit angegriffen werden kann. Dadurch kann die jeweilige Schwachstelle mit Updates geschlossen werden, sofern welche vorhanden sind. Mithilfe von des Hinweises *port* als auch mit *analyzed-port* können z.B. Firewallregeln erstellt werden, die einen jeweiligen Port blockieren, damit ein Vorfall nicht auftreten kann.

Die Hinweise *software-link*, *vendor*, *vendor-homepage*, *website* und *additional-urls* sind URLs die nicht als bösartig zu bewerten sind, da sie als solches keinen Hinweis auf einen Vorfall geben können. Gleiches gilt für Hinweise, die als Kontaktmöglichkeiten zu dem jeweiligen Autor oder der Autorin des Exploits einzustufen sind. Dazu zählen *contact*, *github*, *email*, *irc* und *additional-emails*. Das Merkmal *group* repräsentiert Gruppen die Exploits entwickeln und ihren Namen als Kommentare hinterlegen. Dadurch kann die Herkunft des Exploits ermittelt werden, nicht aber als Hinweis zur Identifizierung eines Vorfalls dienen. Der Hinweis *reference* enthält unterschiedliche Informationen. Dies kann ein Verweis auf einen anderen Exploit, eine CVE oder z.B. einer Herstellerwarnung wie einer SAP Security Note sein und ergeben somit keine nützlichen Informationen zu einem Vorfall. *tested-version* und *version* geben an, welche Version einer Soft- oder Hardware eine Schwachstelle besitzt für die es den jeweiligen Exploit gibt. Daher können diese als einen möglichen Hinweis zu einem Vorfall dienlich sein. Das Merkmal *tested-on* gehört ebenfalls dazu, da es angibt auf welchem Betriebssystem oder welcher Hardware der Exploit Verwendung findet.

Die Merkmale *steps-to-reproduce*, *example* sowie *usage* geben Aufschluss darüber wie eine Schwachstelle ausgenutzt werden kann bzw. wie der Exploit zu verwenden ist. Zur Erkennung oder Vorbeugung eines Vorfalls können diese daher nicht eingesetzt werden. Ein *google-dork* ist eine spezifische Google-Abfrage mit der Informationen über eine Webseite gefunden werden kann. Die Informationen können Wissen über eingesetzte Webtechnologien aufzeigen und folglich, ob sie eine Schwachstelle haben. Ebenso können mit einigen Abfragen sensible Informationen, wie z.B. Benutzernamen oder Passwörter aus Log-Dateien ermittelt werden [81]. In diesem Kontext können sie eingesetzt werden, um zu prüfen, ob der jeweilige Exploit gegen das eigene System einsetzbar ist, da dieser für

eine existierende Schwachstelle auf einer Webseite konzipiert wurde. Daher sind sie ein hilfreiches Werkzeug, um eine Webseite zu untersuchen und etwaige Schwachstellen zu schließen, sodass es zu keinem Vorfall kommen kann.

Weiterhin werden MD5-Hashes aus den Exploits extrahiert. Hashes von Schadprogrammen sind häufige Indikatoren [4]. Die aus dem Exploitcode extrahierten MD5-Hashes können ebenfalls als Hinweise für mögliche Vorfälle angesehen werden. Allerdings sind dies keine Hashes von Schadprogrammen, sondern u.a. von Dynamic-Link Libraries (DLL) oder Programmen die eine Schwachstelle aufweisen. Dadurch ist bspw. ersichtlich, welches Programm eine oder mehrere Schwachstellen aufweist, die der jeweilige Exploit ausnutzt. Die Merkmale *user-agent*, *payload* und *content-type* können gleichermaßen als Hinweise dienlich sein. Beispielsweise können sie in einer Log-Datei auf einem Server aufgezeichnet werden und somit einen Exploit identifizieren. Der Hinweis *help-parser-arg* zeigt den Hilfetext der zu übergebenen Argumente eines Python-Exploits an. Dieser gibt Aufschluss wie der Exploit auszuführen ist und kann aufgrund dessen nicht als Hinweis zur Erkennung von Vorfällen dienen. Zudem wurden die von den Exploits importierten Python-Bibliotheken extrahiert und werden unter dem Merkmal *py-imports* abgespeichert. Sie können für weitere Analysen verwendet werden, wie z.B. zum Aufbau von Graphen die die Abhängigkeiten zu den Bibliotheken aufzeigen. Zur Feststellung von Vorfällen sind sie allerdings nicht dienlich.

Die aus den Exploits ermittelten CVE-Nummern geben weitere Informationen zu der jeweiligen Schwachstelle die ein Exploit ausnutzt. Zudem gibt es Exploits mit mehreren Nummern, die auf andere Schwachstellen verweisen, um weitere Informationen zur Verfügung zu stellen. Zur Identifizierung eines konkreten Vorfalls sind die Nummern indes nicht hilfreich. Weiterhin sind IP-Adressen ein häufiger Indikator zur Erkennung von möglichen Vorfällen [4]. Die in den Exploits identifizierten IP-Adressen können nicht als solche angesehen werden, da sie die Zieladressen der Anzugreifenden darstellen und nicht die Quelladressen von dem ein Angriff ausgeführt wird. Außerdem enthalten 810 Exploits zwischen den Jahren 2009 und 2018 insgesamt 1185 IP-Adressen. Davon sind 216 IP-Adressen aus dem privaten Adressbereich die auf 183 Exploits abfallen. Private Adressen sind u.a. aus dem Bereich 192.168.0.0 bis 192.168.255.255 [53]. Im Zeitraum zwischen 1988 und 2008 gibt es 142 IP-Adressen in 111 Exploits. 22 davon sind private Adressen die in 18 Exploits vorkommen.

Hinweis	Anzahl Hinweise / Anzahl Exploits 2009-2018	Anzahl Hinweise / Anzahl Exploits 1988-2008	Summe Hinweise / Summe Exploits 1988-2018	precursor	indicator
id	2232/2232	254/254	2486/2486	✗	✗
file	2232/2232	254/254	2486/2486	✗	✗
description	2232/2232	254/254	2486/2486	✗	✗
date	2232/2232	254/254	2486/2486	✓	✓
author	2232/2232	254/254	2486/2486	✗	✗
type	2232/2232	254/254	2486/2486	✓	✓
platform	2232/2232	254/254	2486/2486	✓	✓
port	261/261	43/43	304/304	✓	✓
software-link	789/783	0/0	789/783	✗	✗
vendor-homepage	604/604	0/0	604/604	✗	✗
vendor	115/111	9/9	124/120	✗	✗
tested-version	102/102	0/0	102/102	✓	✓
version	1062/1034	6/6	1068/1040	✓	✓
tested-on	1048/1037	11/11	1059/1048	✓	✓
steps-to-reproduce	85/85	0/0	85/85	✗	✗
google-dork	27/27	0/0	27/27	✓	✓
usage	151/143	17/17	168/160	✗	✗
example	35/34	3/3	38/37	✗	✗
reference	33/30	3/3	36/33	✗	✗
md5	27/20	0/0	27/20	✓	✓
contact	125/125	4/4	129/129	✗	✗
github	19/19	0/0	19/19	✗	✗
website	114/109	1/1	115/110	✗	✗
email	82/82	2/2	84/84	✗	✗
irc	16/9	1/1	17/10	✗	✗
analyzed-port	227/225	38/38	265/263	✓	✓
user-agent	68/67	3/3	71/70	✓	✓

Hinweis	Anzahl Hinweise / Anzahl Exploits 2009-2018	Anzahl Hinweise / Anzahl Exploits 1988-2008	Summe Hinweise / Summe Exploits 1988-2018	precursor	indicator
payload	130/107	4/3	134/110	✓	✓
content-type	158/151	13/12	171/163	✓	✓
help-parser-arg	504/121	22/8	526/129	✗	✗
py-imports	4652/1433	618/205	5270/1638	✗	✗
cve	447/373	14/11	461/384	✗	✗
ip	1185/810	142/111	1327/921	✗	✗
additional-urls	4884/1868	254/152	5138/2020	✗	✗
additional-emails	557/447	54/50	611/497	✗	✗
group	163/163	181/181	344/344	✗	✗

Tabelle 6.1
Analyseergebnisse untersuchter Python-Exploits sowie Einteilung der Hinweise in Kategorien innerhalb drei angegebener Zeiträume

Tabelle 6.2 summiert, auf Basis von Tabelle 6.1, die ermittelten Hinweise aus der Anzahl der Exploits in denen sie vorkommen aus beiden Zeiträumen. Zudem ist die Anzahl der Hinweise die der *precursor*- und *indicator*-Kategorie zugewiesen oder nicht zugewiesen wurden, aufsummiert dargestellt. Die Zeilen der Tabelle unterteilen die Herkunft der Hinweise. Die erste Zeile bildet die Hinweise die in der CSV-Datei vorkommen ab. Die zweite Zeile hingegen zeigt die aus der Analyse ermittelten Hinweise. In der dritten Zeile werden diese Werte nochmals zusammen addiert.

Dies verdeutlicht, wie viele Hinweise bereits durch die vorgegebenen Metadaten existieren und welche durch die Analyse von Kommentaren und Quellcode hinzu gekommen sind. Zwischen den Jahren 2009 und 2018 sind demnach ca. 52,29% der Informationen aus der Analyse zu den bestehenden CSV-Metadaten hinzugekommen. Dies entspricht dem Verhältnis von 17.409 zu 33.294. Im Zeitraum von 1988 bis 2008 konnten ca. 43,46% zusätzliche Informationen durch die Analyse ermittelt werden und kommt dem Verhältnis

von 1400 zu 3221 gleich. Über den gesamten Zeitraum aller betrachteten Python-Exploits sind dies ca. 51,51% (18.809 zu 36.515).

Herkunft	**Summe Hinweise / Anzahl Exploits 2009-2018**	**Summe Hinweise / Anzahl Exploits 1988-2008**	**Summe Hinweise / Anzahl Exploits 1988-2018**	**Summe precursor**	**Summe indicator**
Aus CSV-Datei	15.885/2232	1821/254	17.706/2486	✓4 ✗ 4	✓4 ✗ 4
Analysiert	17.409/2232	1400/254	18.809/2486	✓9 ✗ 19	✓9 ✗ 19
Summe	33.294/2232	3221/254	36.515/2486	✓13 ✗ 23	✓13 ✗ 23

Tabelle 6.2
Summe der ermittelten Hinweise und ihrer Kategorisierung innerhalb von drei Zeiträumen

Tabelle 6.3 zeigt zum einen die Anzahl der Hinweise die in die beiden Kategorien *precursor* und *indicator* eingeteilt wurden. Zum anderen zeigt sie die Summe der ermittelten Hinweise in den beiden Zeiträumen. Zudem wird zwischen den Hinweisen differenziert die bereits durch die CSV-Metadaten vorhanden sind und denjenigen die aus den betrachteten Python-Exploits extrahiert wurden. Daraus ist zu erkennen, dass zwischen den Jahren 2009 und 2018 bereits 6957 verwendbare Hinweise, ca. 70,95%, durch die vorhandene CSV-Datei zur Verfügung stehen. Dem gegenüber stehen 2849 verwendbare Hinweise, ca. 29,05%, aus dem Analyseprozess. Wird diese Summe von 9806 Hinweisen ins Verhältnis zu der Gesamtsumme aller ermittelten Hinweise von 33.294 gestellt (siehe Tabelle 6.2), ergibt sich ein Anteil von 29,45% brauchbaren Hinweisen aus der Analyse von 2232 Python-Exploits. Im Zeitraum zwischen den Jahren 1988 und 2008 wurden ca. 27,32% brauchbare Hinweise ermittelt.

Zusammenfassend können ca. 30% der ermittelten Hinweise in die *precursor*- und *indicator*-Kategorien eingeteilt werden. Dieser Wert kann an der Stelle nicht bewertet werden, da nicht bekannt ist, ob die Hinweise in der Praxis hilfreich sind. Dazu müssten die Zielgruppen die gewonnenen Hinweise auf ihren Nutzen hin bewerten.

	Anzahl der ✓-Hinweise	**Summe der Hinweise 2009-2018**	**Summe der Hinweise 1988-2008**
Nur CSV-Metadaten	4	6957	805
Ohne CSV-Metadaten	9	2849	75
Summe	**13**	**9806**	**880**

Tabelle 6.3
Gegenüberstellung der in die Kategorien eingeordneten Hinweise mit und ohne CSV-Metadaten in zwei angegebenen Zeiträumen

6.1.2 Eingliederung der Informationen (FF02)

Frage: In welche der vier Kategorien von CTI (Strategic Threat Intelligence, Operational Threat Intelligence, Tactical Threat Intelligence, Technical Threat Intelligence), nach Chismon und Ruks [4] aus Abschnitt 2.1, können die gewonnenen Informationen eingegliedert werden?

Hypothese 1: Wenn Hinweise aus der *precursor*- und *indicator*-Kategorie aus diesen Exploits gewonnen werden können, dann gehören sie zu den Technical Threat Intelligence.

Hypothese 2: Wenn die gewonnenen Informationen Aufschluss über das Vorgehen von Angreifenden geben, dann zählen sie zu den Tactical Threat Intelligence.

Wie aus dem vorherigen Abschnitt hervorgeht, konnten aus den analysierten Python-Exploits erfolgreich Hinweise aus der *precursor*- sowie *indicator*-Kategorie gewonnen werden. Daher wird die Hypothese 1 erfüllt und die Hinweise gehören zu den Technical Threat Intelligence. Weiterhin kann ausgeschlossen werden, dass die Informationen sowohl zu den Strategic als auch zu den Operational Threat Intelligence gehören. Zunächst sind es technische Informationen, sodass es keine Strategic TI sein können. Ferner können daraus keine Risiken erkannt und keine Entscheidungen getroffen werden, da die extrahierten Hinweise nicht auf bevorstehende Angriffe hindeuten können. Letztlich geben sie Andeutungen auf die technische Ausnutzung von Schwachstellen. Dieses Argument untermauert zudem, dass die Hinweise nicht zu den Operational TI gezählt werden können. Des Weiteren sind die extrahierten Kontaktmöglichkeiten - wie *email*, *contact* oder *website* - keine hilfreichen Merkmale zur Generierung von Operational TI durch

Überwachung dieser Kanäle. Dabei handelt es sich nicht um die Kontakte der Angreifenden, sondern die der Forschenden die den jeweiligen Exploit entwickelt haben.

Tactical Threat Intelligence werden im Allgemeinen als Tactics, Techniques and Procedures (TTPs) bezeichnet und geben Aufschluss über das Verhalten von Angreifenden. Ein paar wenige der durch die Exploit-Analyse extrahierten Indikatoren können dennoch dazu gezählt werden, sodass diese als Tactical TI zu betrachten sind. Der folgende Absatz zeigt einige Beispiele auf.

Mithilfe einiger extrahierter Payloads können Instruktionen zur Ausführung von Anweisungen identifiziert werden. Diese können möglicherweise Aufschluss über das Vorgehen zur Kompromittierung eines Systems geben und können dadurch unter den Begriff *Techniques* gefasst werden. Der Indikator *google-dork* kann hingegen unter den Begriff *Tactics*, zum Sammeln anfänglicher Informationen von Angreifenden, aufgefasst werden. Die dadurch ermittelten Informationen können zur weiteren Planung zukünftiger Angriffe und den Einsatz des jeweiligen Exploits dienlich sein. Zusätzlich kann das in MISP erstellte Exploit-POC-Object manuell analysiert werden, um weitere technische Hinweise zum Vorgehen der Angreifenden zu gewinnen. Daher ist es unter den Begriff *Techniques* einzugliedern.

Grundsätzlich bedingt es einer zusätzlichen Analyse der Indikatoren bzw. des Exploit-POC-Objects, um Tactical Threat Intelligence zu generieren.

6.1.3 Verwendung der Informationen (FF03)

Frage: Welche Zielgruppen können die gewonnenen Informationen aus den Exploits verwenden?

Hypothese: Wenn die Informationen als CTI zu betrachten sind (siehe FF02), dann können alle in Kapitel 3 genannten Zielgruppen diese für die eigenen Ziele nutzen.

Wie aus der Beantwortung der Forschungsfrage FF02 hervorgeht, werden aus der Exploit-Analyse Technical und Tactical Threat Intelligence produziert. Daher werden zur Beantwortung der hier gestellten Forschungsfrage und deren Hypothese nur diese betrachtet. Weiterhin wurde bereits erwähnt, dass die Exploit-Analyse als eine weitere Quelle von CTI zu sehen ist. Daher werden die Zielgruppen hier als Konsumenten dieser betrachtet. Des Weiteren wurde in Kapitel 3 bereits festgestellt, dass alle Zielgruppen sowohl Technical als auch Tactical Threat Intelligence beziehen. Aus diesen Folgerungen ist die

Hypothese vollständig belegt. Zudem können alle Zielgruppen die gewonnenen Informationen aus der Analyse der Exploits verwenden.

6.1.4 Zusammenhang zwischen Schwachstellen und Exploits (FF04)

Frage: Welche Aussage kann zwischen der Anzahl veröffentlichter Schwachstellen und der Anzahl veröffentlichter Exploits getroffen werden?

Hypothese: Wenn es eine bestimmte Anzahl veröffentlichter Schwachstellen pro Jahr gibt, dann gibt es, relativ gesehen dazu, viele veröffentlichte Exploits im selben Jahr.

Abbildung 6.1 zeigt die Anzahl der in der ExploitDB vorhandenen Exploits sowie die Anzahl der in der National Vulnerability Database (NVD) veröffentlichten Schwachstellen zwischen den Jahren 2009 und 2018. Die Balken repräsentieren die Exploits und die Kurve die Schwachstellen. Für eine bessere Übersicht sind die jeweiligen Werte pro Jahr in der Legende oben links in der Abbildung angegeben.

Es ist ersichtlich, dass jedes Jahr, mit Ausnahme von 2010, erkennbar mehr Schwachstellen verzeichnet als Exploits öffentlich zugänglich gemacht wurden. Mögliche Gründe für den Anstieg vorhandener Schwachstellen wurden bereits zu Beginn von Abschnitt 4.1 erläutert. Auffallend sind die Jahre 2017 und 2018 in denen deutlich mehr Schwachstellen als Exploits vorhanden sind. Es ist insgesamt zu erwarten, dass die Anzahl der veröffentlichten Exploits mit der Steigerung der Veröffentlichung von Schwachstellen ebenso ansteigt. Dies ist aus der Abbildung nicht erkennbar, sodass die obige Hypothese nicht zutreffend ist.

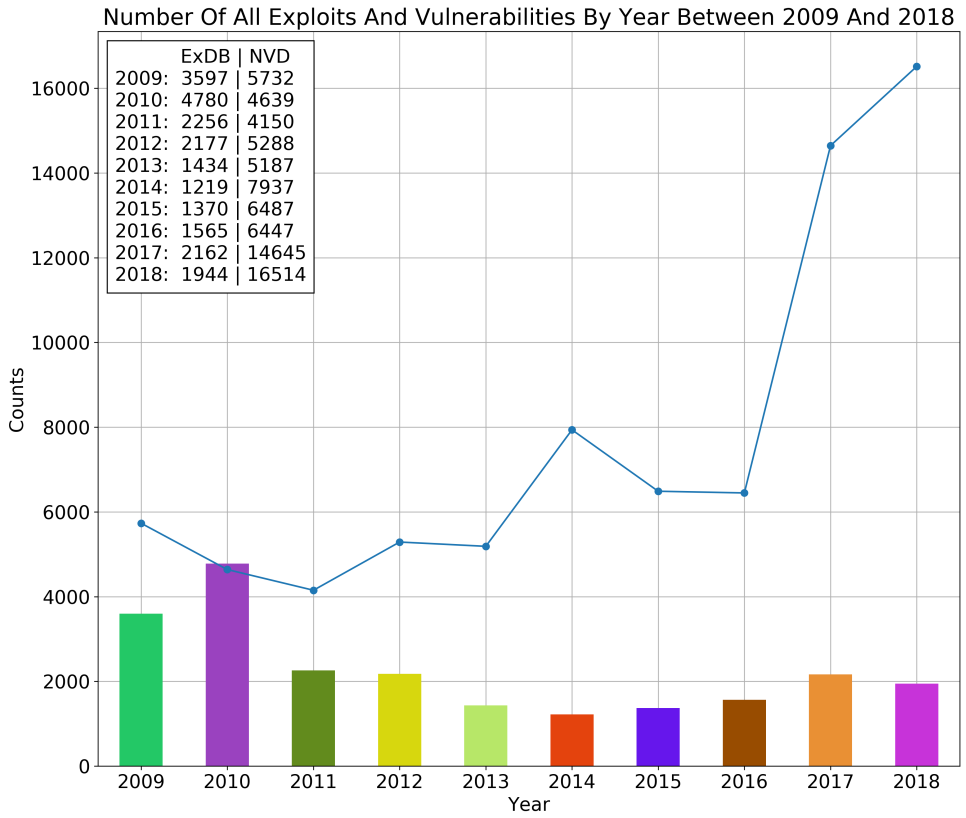

Abbildung 6.1
Anzahl der Exploits und Schwachstellen zwischen den Jahren 2009 und 2018 [61]

Letztlich kann nur bedingt ein Zusammenhang zwischen den veröffentlichten Schwachstellen und Exploits hergestellt werden. Zum einen bekommt jede Schwachstelle in der NVD eine CVE-Nummer zugewiesen. Ein Exploit hingegen nutzt nicht zwangsläufig eine Schwachstelle mit einer CVE-Nummer aus. Dies bedeutet, dass Exploits öffentlich zugänglich sein können, ohne dass eine Schwachstelle in der Datenbank vorhanden ist. Dies ist in dem Jahr 2010 sehr gut erkennbar. Unter diesem Gesichtspunkt ist dabei eine Verzerrung der Statistik unumgänglich. Zum anderen stammen die hier aufsummierten Daten aus unterschiedlichen Quellen, die unterschiedliche Ziele verfolgen. Infolgedessen ist ein Vergleich ohne direkten Zusammenhang nicht möglich. Wünschenswert ist eine Datenbasis mit Schwachstellen und Exploits die eine Eins-zu-Eins-Beziehung unterein-

ander besitzen. Damit kann die Erwartung, dass die Anzahl der bereitgestellten Exploits mit der Anzahl der veröffentlichten Schwachstellen ebenso ansteigt, vielmehr bewertet werden.

6.1.5 Zeitlicher Zusammenhang zwischen CVE und Exploits (FF05)

Frage: Kann ein zeitlicher Zusammenhang zwischen der Veröffentlichung einer neuen CVE und einem Exploit hergestellt werden?

Hypothese: Wenn eine neue CVE veröffentlicht wurde, dann folgt kurze Zeit später die Veröffentlichung eines dazugehörigen Exploits.

Es kann durchaus ein zeitlicher Zusammenhang zwischen der Veröffentlichung einer neuen CVE und einem dazugehörigen Exploit hergestellt werden. Dies sehe vor, das jeweilige Datum der Veröffentlichung miteinander zu vergleichen. Jedoch ist dies derzeit nicht automatisiert umsetzbar, da die Daten aus der ExploitDB kein Datum für CVE-Nummern enthält. Daher müsste die Analysesoftware um eine Komponente erweitert werden, die das Datum der CVEs bspw. aus der NVD ermittelt und diese mit dem Datum der Veröffentlichung des korresponierenden Exploits vergleicht.

Eine manuelle Lösung ist allerdings mithilfe der MISP-Oberfläche möglich. Sobald ein Attribut mit einer CVE-Nummer erstellt wurde, werden mithilfe des CVE-Modules die Informationen zu einer CVE abgerufen und das Event dadurch angereichert [34]. Dadurch kann das Erstellungsdatum des Exploits mit dem der CVE-Nummer innerhalb der Oberfläche verglichen werden. In Anbetracht von 373 Exploits die zwischen den Jahren 2009 und 2018 veröffentlicht wurden und mindestens eine CVE-Nummer enthalten, ist das ein hoher Aufwand dies manuell zu lösen, um die Hypothese zu beantworten. Jedoch zeigt die Evaluation des Anwendungsfalls im Unterkapitel 6.5 die Tendenz auf, dass in der Regel zuerst die Schwachstellen veröffentlicht werden und anschließend ein Exploit zur Verfügung steht. Dabei ist eine hohe Schwankung der Differenzen des jeweiligen Datums einer Schwachstelle und eines Exploits zu sehen. Zum einen wurden Exploits kurz nach der Veröffentlichung einer CVE bereitgestellt. Zum anderen können Monate oder Jahre dazwischen liegen. Dies kann jedoch nicht als eine ausreichende Analyse zur stichhaltigen Beantwortung der Hypothese herangezogen werden.

6.1.6 Plattformverfügbarkeit der Exploits (FF06)

Frage: Für welche Plattform stehen die meisten Exploits zur Verfügung?

Hypothese: Je populärer eine Plattform ist, desto eher ist sie Ziel von Angriffen durch Exploits.

In Abbildung 6.2 sind die meist genutzten Plattformen zwischen den Jahren 2009 und 2018 als Tortendiagramm abgebildet. Plattformen umfassen jegliche Systeme, ob Software oder Hardware, auf die ein Exploit Anwendung findet. In diesem Zeitraum sind 22.504 Exploits verzeichnet mit insgesamt 46 unterschiedlichen Plattformen. Daraus ist zu erkennen, dass PHP mit knapp 50% die Plattform mit den meisten Exploits darstellt. PHP-Exploits sind u.a. jene die PHP-Code ausführen oder durch einen HTTP-Request Schwachstellen in PHP-Funktionen ausnutzen. Darauf folgt Windows mit 25,12%, Multiple mit 6,55% und Linux mit 5,62%. Windows-Plattformen werden in der ExploitDB zusätzlich differenziert. Daher gibt es die Bezeichnungen *windows*, *windows_x86* und *windows_x86-64*. Sie werden in den hier aufgeführten Statistiken ebenfalls separat behandelt. Multiple-Plattformen umfassen mehrere Plattformen gegen die ein Exploit eingesetzt werden kann.

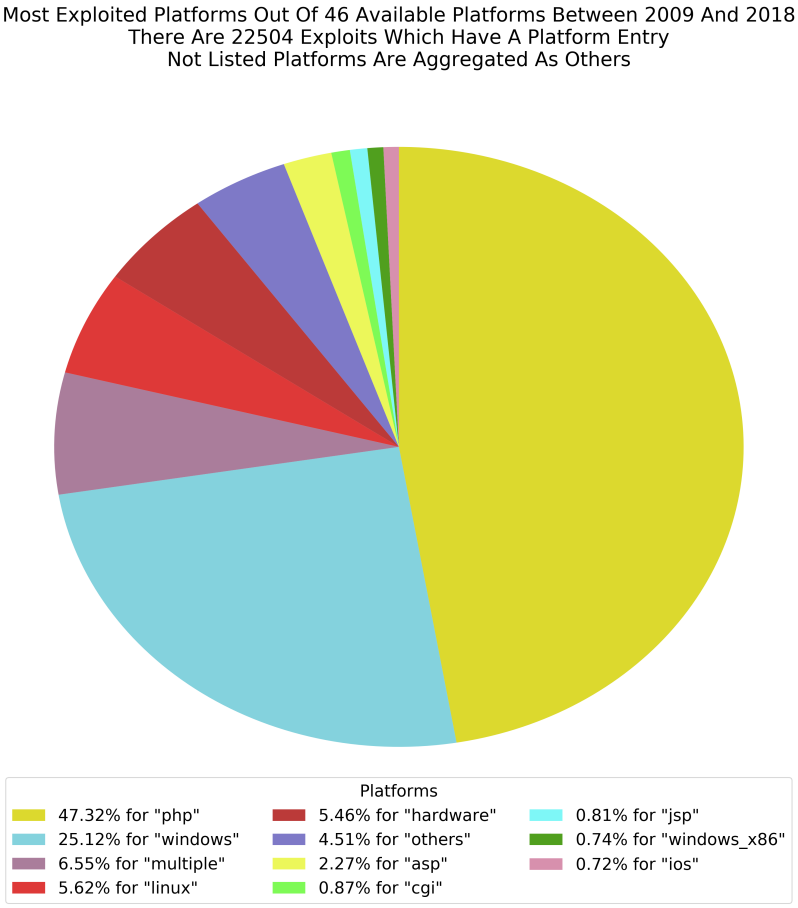

Abbildung 6.2
Zehn meist ausgenutzte Plattformen zwischen den Jahren 2009 und 2018

Die aufgestellte Hypothese kann nicht eindeutig beantwortet werden. Das liegt daran, dass es keine umfassende Statistik gibt, die die selben Plattformen über den gleichen Zeitraum aggregiert wie die hier aufgezeigte. Weiterhin wird PHP im Allgemeinen nicht als eine Plattform aufgefasst, sogleich sie hier diejenige mit den meisten Exploits ist. In der ExploitDB sind zudem Exploits für mobile Plattformen wie Android oder iOS enthalten. StatCounter [89] zeigt den Verlauf der Beliebtheit von Desktop, mobilen und Tablet Betriebssystemen zwischen 2009 und 2018 auf. Daraus ist ersichtlich, dass Windows jahrelang das meist genutzte Betriebssystem war. Mittlerweile hat Android diesen Platz eingenommen. In Abbildung 6.2 kommt Android allerdings nicht unter den zehn häufigsten Plattformen, für die es Exploits gibt, vor. Allerdings wurden zwischen 1988

und 2018 insgesamt 122 Android-Exploits in der ExploitDB bereitgestellt. Davon fallen 120 auf den Zeitraum zwischen 2009 und 2018 ab.

Abschließend kann nur vage ausgesagt werden, dass eine populäre Plattform eher ein Ziel von Angriffen durch Exploits ist. PHP ist wie bereits erwähnt keine gängige Plattform, wird aber als solche innerhalb von ExploitDB definiert. Windows hingegen ist weiterhin populär und ist ebenso ein verbreitetes Ziel für Angriffe mithilfe von Exploits aus der ExploitDB. Multiple-Plattformen können an dieser Stelle nicht unterschieden werden, da nur innerhalb des Exploits dokumentiert ist, für welche Systeme dieser zutrifft. Des Weiteren gibt es bisher sehr wenige Exploits in der ExploitDB für Android, obwohl dieses das mittlerweile meist genutzte System ist. Dies kann sowohl bedeuten, dass es ein sehr sicheres System ist, als auch, dass es noch nicht in den Fokus für die Entwicklung von Exploits innerhalb des Kreises von Forschenden gerückt ist.

6.1.7 Verwendete Programmiersprachen (FF07)

Frage: In welcher Programmiersprache werden die meisten Exploits verfasst?

Hypothese: Je populärer eine Programmiersprache ist, desto eher wird diese zur Erstellung von Exploits eingesetzt.

Abbildung 6.3 zeigt das Tortendiagramm mit den meist genutzten Dateiendungen zur Erstellung von Exploits zwischen 2009 und 2018. Es wurden in 22.504 Exploits 41 unterschiedliche Endungen gefunden. Die Endung *txt* wurde am häufigsten genutzt. Diese umfassen PoCs die die Ausnutzung von Schwachstellen in Textform beschreiben. Die Beschreibungen umfassen u.a. die Bereitstellung von Codebeispielen die injiziert werden müssen, oder einer URL mit detaillierteren Informationen zu dem Exploit. Darauf folgen die Endungen für Python mit 10,69%, Ruby mit 9,19%, Perl mit 4,53%, HTML mit 4,5% und C mit 3,19%. Zusätzlich zeigt das Balkendiagramm in Abbildung 6.4 den prozentualen Anteil der häufigsten Endungen (ohne *txt*) pro Jahr zwischen 2009 und 2018. Für eine bessere Vergleichbarkeit, wurde die Endung *txt* ausgeblendet. Der absolute Anteil von Exploits in Textform spiegelt sich jedoch in den relativen Anteilen der angegebenen Endungen wider. Die in Textform verfassten Exploits verlaufen konstant über die Jahre, bei um die 60%, im Verhältnis zu allen anderen. Wohingegen die in Python verfassten Exploits mit den Jahren anteilig zugenommen hat. Ruby dagegen erfuhr einen kurzen Höhepunkt bis 2013 und wurde anschließend weniger häufig eingesetzt.

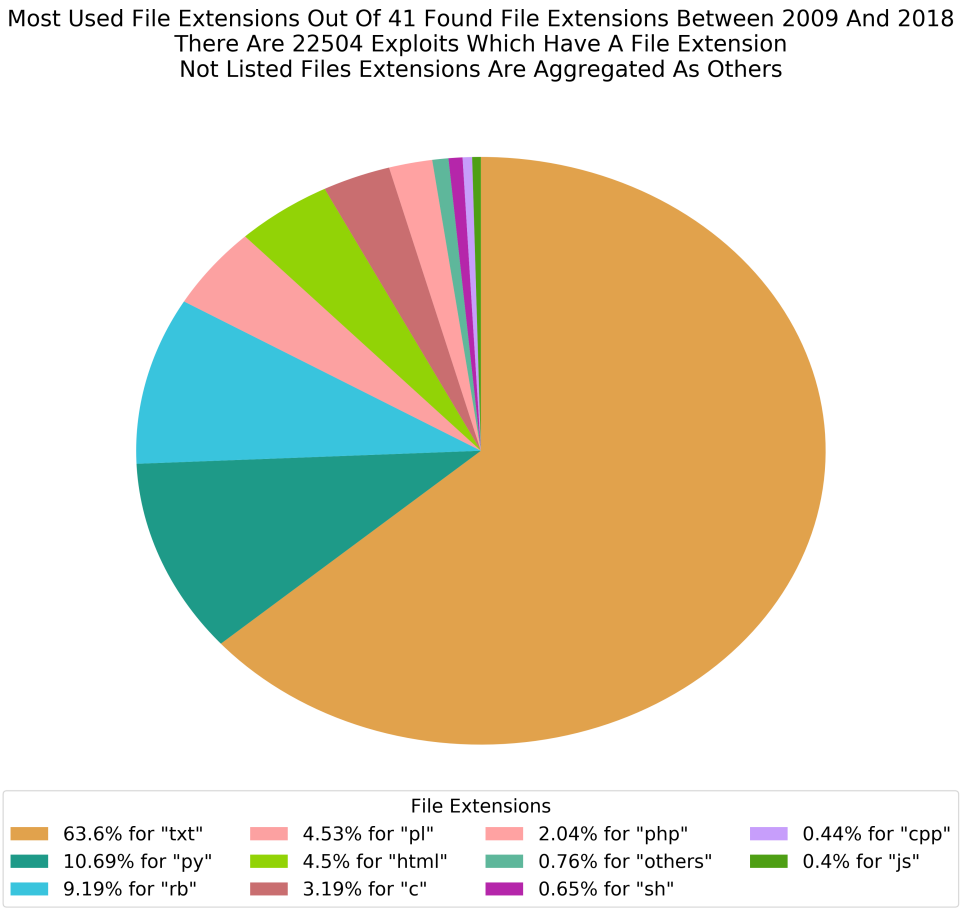

Abbildung 6.3
Zehn meist genutzte Dateiendungen zwischen den Jahren 2009 und 2018

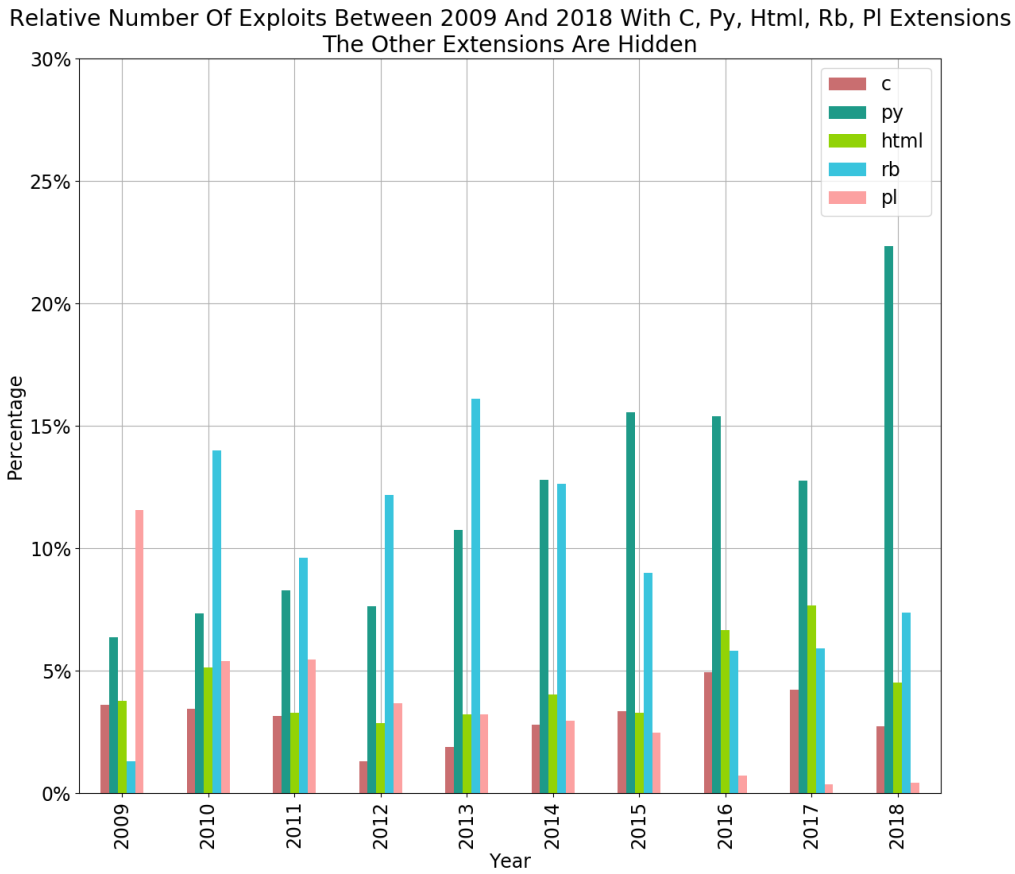

Abbildung 6.4
Meist genutzte Dateiendungen (ohne *txt*) der Exploits in Prozent aufgeteilt nach Jahren zwischen 2009 und 2018

Das Video „Most Popular Programming Languages 1965 - 2019“ [9] zeigt die elf populärsten Programmiersprachen zwischen den Jahren 1965 bis 2019 in Quartalen pro Jahr in einer Zeitleiste. Die zugrundeliegenden Daten stammen von GitHub, aus anderen nicht näher genannten aggregierten Statistiken sowie aus verschiedenen Studien. Zum Vergleich mit den meist genutzten Sprachen zur Entwicklung von Exploits, werden aus den Ergebnissen des Videos, jeweils die letzten Quartale eines Jahres, aus dem Zeitraum zwischen 2009 und 2018, herangezogen. Diese stellen die Aggregation der Popularität von Programmiersprachen pro Jahr dar.

Darin ist zunächst zu erkennen, dass Java und JavaScript die beiden beliebtesten Spra-

chen in dem Zeitraum 2009 bis 2017 sind. Zum Ende des Jahres 2018 schließt Python auf und nimmt den ersten Platz knapp vor Java ein. Java bleibt innerhalb dieses Zeitraums kontinuierlich vor JavaScript. Wohingegen der Abstand zunächst bei mehr als 10% liegt. JavaScript schließt jedoch zum Ende der Zeitspanne zu Java auf. Python startet im Jahr 2009 von Platz sechs, hinter C und C++. Bereits zwei Jahre später überholt Python diese beiden Sprachen und wird indes populärer. C und C++ befinden sich ab diesem Zeitpunkt durchgehend auf Platz sieben bzw. sechs. Ruby dagegen schafft es in dem Zeitraum nie unter die beliebtesten acht Sprachen und verbleibt durchgehend zwischen den Plätzen neun und elf. Die Programmiersprache Perl ist in diesem Zeitraum nicht unter den beliebtesten elf Sprachen zu finden.

Zudem gibt es diverse Plattformen die mithilfe unterschiedlicher Merkmale die Popularität von Sprachen zu einem einzelnen Jahr messen. Eine davon ist der „Stack Overflow Annual Developer Survey“ [88]. Diese Onlineumfrage findet einmal jährlich statt. Darin können Entwickler*innen ihre Erfahrungen zu diversen Aspekten in der Software Entwicklung mitteilen. Auf der einen Seite wird nach der meist geschätzten Programmiersprache gefragt. Demnach liegt zwischen den Jahren 2016 und 2019 die Sprache Rust auf Platz eins. Python befindet sich durchgehend unter den ersten zehn bei einer Beliebtheit von ca. 62% bis ca. 73%. Ruby befindet sich über die Jahre hinweg bei ca. 50%.

Auf der anderen Seite wird in der Studie nach den Programmiersprachen gefragt nach denen am ehesten gesucht wird. Python liegt hierbei zwischen den Jahren 2017 und 2019 kontinuierlich auf Platz eins. 2015 und 2016 liegt sie bereits unter den ersten vier. Ruby liegt indes deutlich weiter hinten in dieser Bewertung.

Generell ist die aufgestellte Hypothese zu widerlegen. Rust gilt als die beliebteste Programmiersprache nach dem „Stack Overflow Annual Developer Survey“ [88]. Dem gegenüber steht, dass in der gesamten Datenmenge der ExploitDB kein einziger Exploit in Rust zu finden ist. Weiterhin sind Java und JavaScript nach der Auswertung von Data Is Beautiful [9] im Zeitraum zwischen 2009 und 2017 die zwei populärsten Sprachen. Beziehungsweise gleichauf mit Python im Jahr 2018. Im selben Zeitraum wird JavaScript mit 0,4% zur Entwicklung von Exploits eingesetzt (siehe Abbildung 6.3). Java hingegen ist nicht unter den zehn meist genutzten Dateiendungen vertreten.

Es gibt jedoch Tendenzen, dass populäre Programmiersprachen eher zur Entwicklung von Exploits eingesetzt werden oder sich dafür besonders eignen. Zum einen wurde Python in den vergangenen zehn Jahren vermehrt zur Entwicklung von Exploits eingesetzt. Die vorgestellten Studien zeigen ebenso auf, dass es eine weit verbreitete und beliebte

Sprache ist. Gleichermaßen trifft dies auf Ruby zu, da der Anteil vorhandener Exploits in Ruby ähnlich hoch ist wie der in Python. Zusätzlich scheint es eine geschätzte Programmiersprache zu sein.

6.2 Auswertung für die Sprache Python

Das in Abbildung 6.5 dargestellte Balkendiagramm stellt die absolute Anzahl der Python-Exploits pro Jahr im Zeitraum zwischen den Jahren 2009 und 2018 dar. Daraus ist ersichtlich, dass 2018 die meisten Exploits in dieser Sprache innerhalb der Dekade veröffentlicht wurden. Zudem sind Python-Exploits die zweit häufigsten im Jahr 2018, mit etwas mehr als 20%, nach den Exploits die in Textform verfasst wurden. Diese werden für eine anschaulichere Vergleichbarkeit jedoch ausgeblendet und liegen über den gesamten Zeitraum bei ca. 60%. Überdies ist der Anteil in diesem Jahr der höchste aus dem betrachteten Zeitraum, wie Abbildung 6.4 illustriert. Das Jahr 2010 verzeichnet die zweit höchste Anzahl veröffentlichter Python-Exploits darin. Dieses Jahr ist allerdings das Jahr mit den insgesamt meisten Exploits. Daher ist der Anteil der Python-Exploits geringer als 10%.

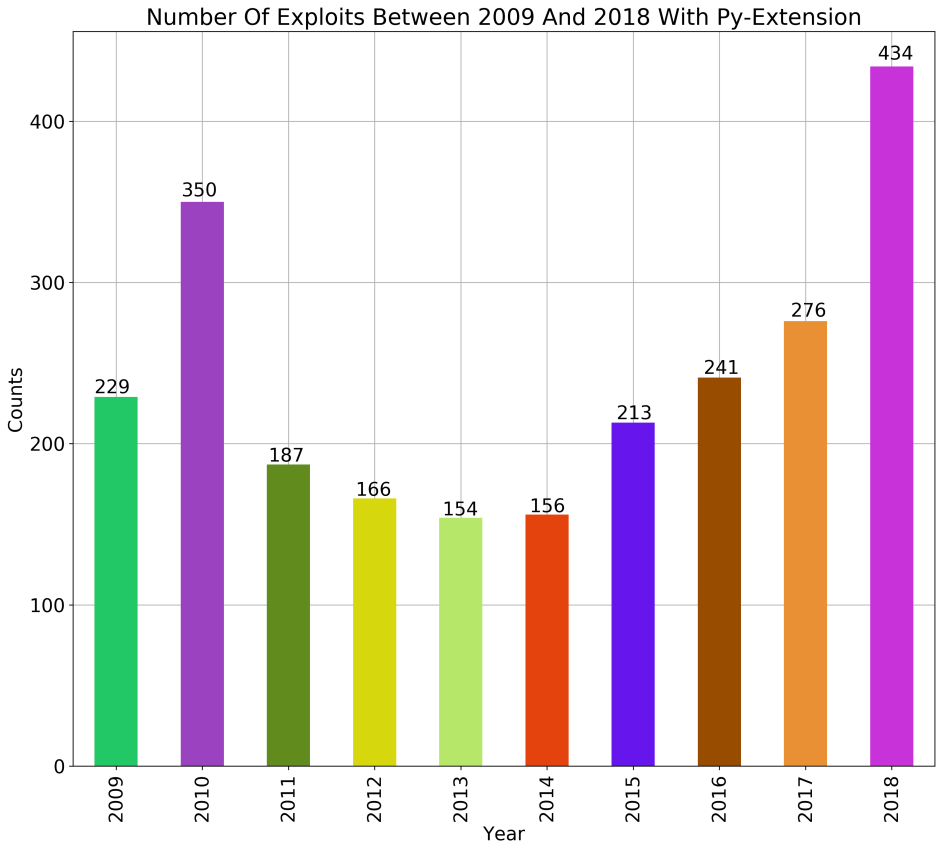

Abbildung 6.5
Anzahl der in Python verfassten Exploits nach Jahren zwischen 2009 und 2018

Abbildung 6.6 zeigt den absoluten Anteil der Python-Exploits zu den jeweiligen Plattformen auf. Demnach fallen mehr als die Hälfte der 2406 Exploits zwischen 2009 und 2018 auf die Plattform Windows ab. Zudem bilden die 1387 Python-Exploits für Windows ein sechzehntel aller Exploits in dem angegebenen Zeitraum ab. Diese Folgerung kann daraus abgeleitet werden, dass ca. ein Viertel aller Exploits für die Plattform Windows anwendbar sind (siehe Abbildung 6.2).

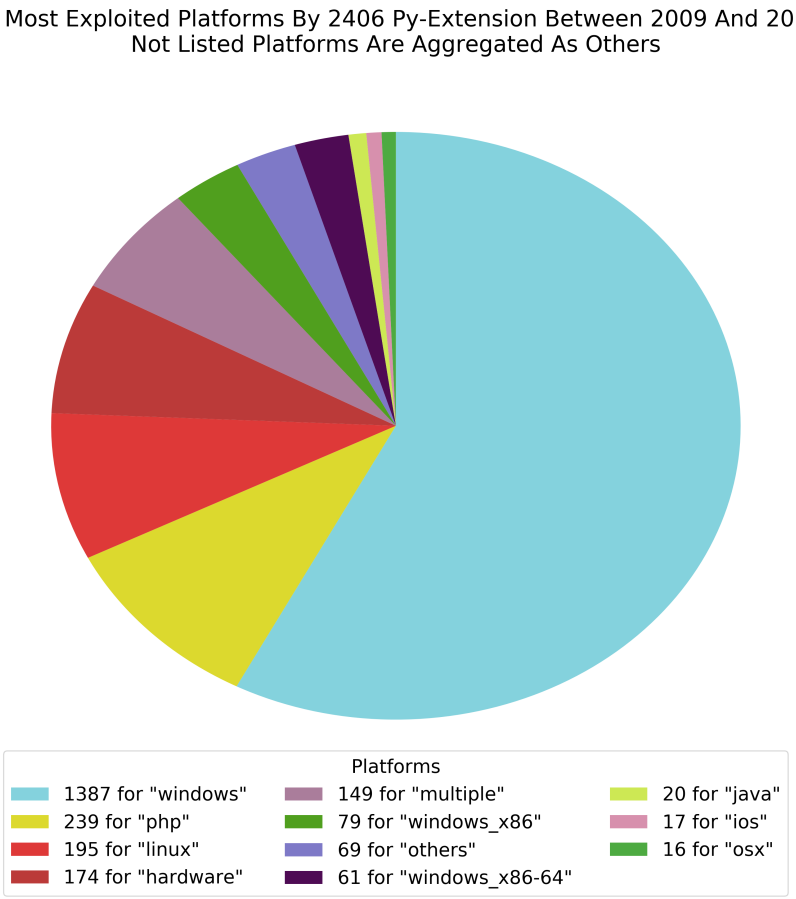

Abbildung 6.6
Zehn meist ausgenutzte Plattformen mithilfe von Python-Exploits zwischen den Jahren 2009 und 2018

Abbildung 6.7 illustriert den relativen Anteil der Exploit Typen aller 2406 Python-Exploits zwischen 2009 und 2018. Demzufolge sind die meisten Exploits mit 35,58% vom Typ *dos*. Exploits der Typen *local* und *remote* sind in etwa zu gleichen Teilen mit knapp ein Viertel vertreten. Python-Exploits für den Typ *webapps* wurden am seltensten entwickelt.

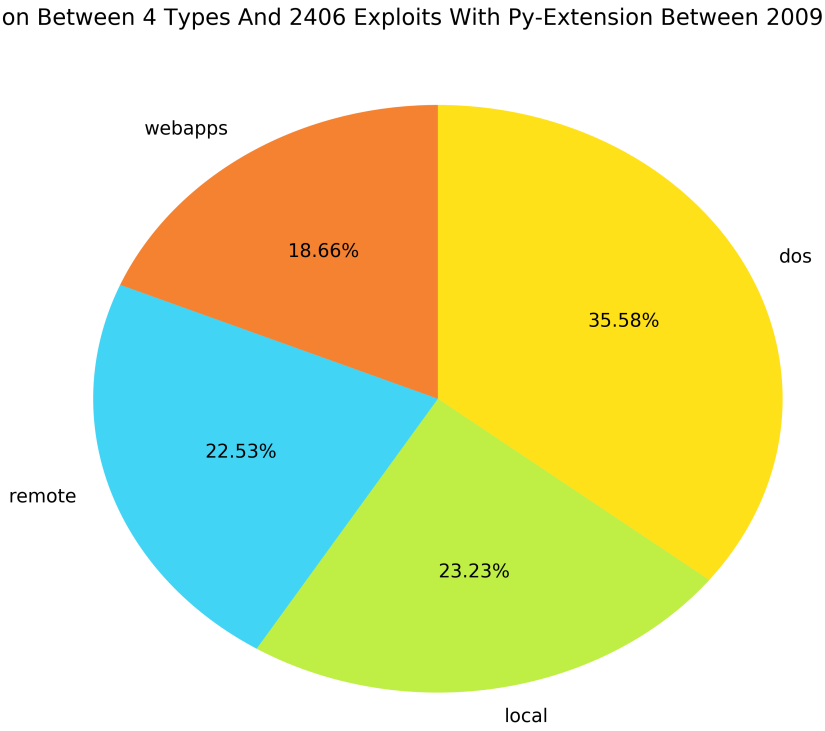

Abbildung 6.7
Verhältnis zwischen den Exploittypen und Python-Exploits zwischen den Jahren 2009 und 2018

6.3 Typen von Sicherheitslücken

Die offizielle Webseite der ExploitDB bietet die Filterung von Exploits mithilfe von sogenannten *Tags* an [63]. Dies sind Typen von Sicherheitslücken wie z.B. Buffer Overflow oder SQL Injection. Abbildung 6.8 zeigt eine eigene Statistik zu diversen Typen nach denen innerhalb der Beschreibungen der Exploits gesucht wurde. Sie stellt den Zeitraum zwischen 2009 und 2018 mit insgesamt 22.504 Exploits dar. In Anlehnung an die offizielle Webseite wurden einige der Typen heraus gesucht. Neben den in der Legende aufgeführten Bezeichnungen wurden außerdem einige Abkürzungen oder andere Schreibweisen

der jeweiligen Art der Sicherheitslücke zur Suche in den Beschreibungen verwendet, wie bspw. “Cross-Site Request Forgery“ und “CSRF“ oder “Denial of Service“ und “DoS“. Dies spiegelt vielmehr eine Tendenz zur Kategorisierung von Sicherheitslücken wider, als ein exaktes Abbild dessen. Eine genauere Statistik wäre möglich, wenn jeder Exploit ein weiteres Metadatum in der CSV-Datei bekäme. Eine weitere Auffälligkeit ist der große Unterschied zu den Ergebnissen jedes Typs dieser Statistik im Gegensatz zu denen auf der offiziellen Webseite. Demnach konnte 2323 mal der Ausdruck “Buffer Overflow“ in den Beschreibungen der Exploits gefunden werden. Auf der Webseite hingegen wurden 229 Exploits, bis zum 04.10.2019, mit diesem Typen versehen. Allerdings ist der älteste Exploit von diesem Typen auf den 29.11.2017 datiert. Grundsätzlich trifft diese Beobachtung auf die anderen Typen ebenso zu. Dies lässt den Schluss zu, dass erst ab einem gewissen Zeitpunkt begonnen wurde die Exploits zu kategorisieren. Wie es zu diesen *Tags* auf der Webseite kommt ist zudem nicht ersichtlich.

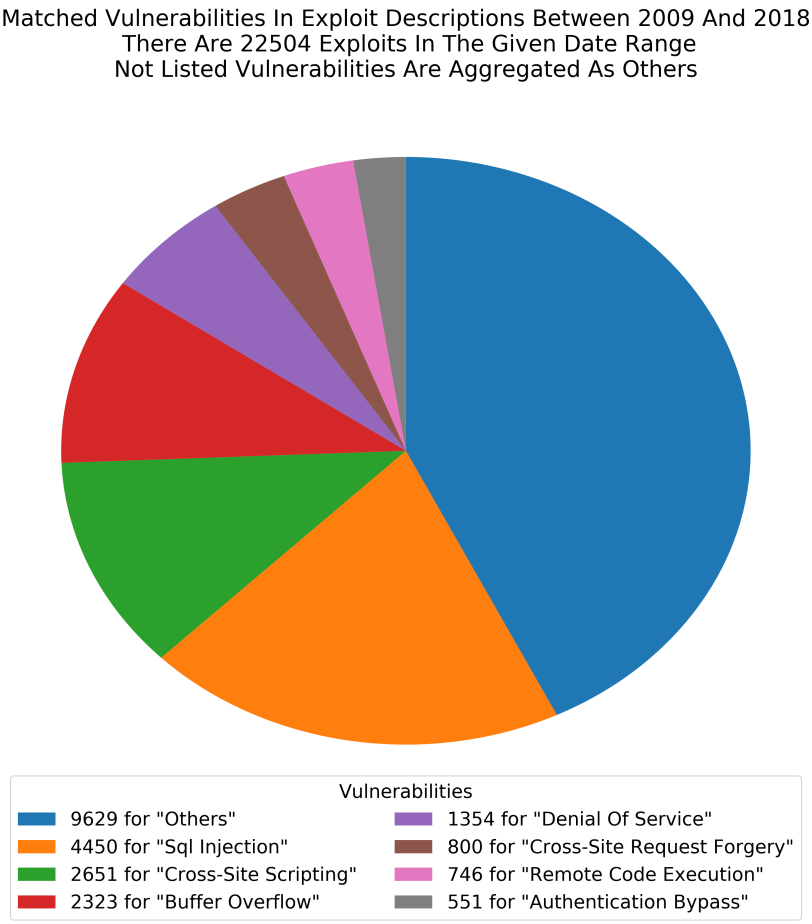

Abbildung 6.8
Gefundene Schwachstellen in den Beschreibungen der Exploits zwischen den Jahren 2009 und 2018

6.4 Hindernisse während der Entwicklung

Während der Konzipierung und Entwicklung der Analysesoftware sind diverse Hindernisse aufgetreten. In diesem Unterkapitel werden einige aufgezeigt. Sofern es eine Lösung gab, wird diese ebenfalls erläutert. Verschiedene Schwierigkeiten wurden bereits in anderen Kapiteln erwähnt und werden hier nur kurz angerissen oder gar nicht ausgeführt.

Zunächst wurde die Analysesoftware in Python 3.5.3 entwickelt. Während der Entwicklung der Codeanalyse, stellte sich schnell heraus, dass nur wenige ASTs aus den jeweiligen

Exploits erzeugt werden konnten. Dies liegt darin begründet, dass Python nur ASTs aus dem vorliegenden Code erzeugen kann, der in der selben Hauptversion verfasst ist wie der ausführende Interpreter. Dies führte zu vielen Exceptions während der Codeanalyse und der Erkenntnis, dass ein hoher Anteil der Exploits in Python 2 verfasst ist. Aus diesem Grund wurde die Analysesoftware in Python 2.7.13 verfasst. Der Umstieg löst das generelle Problem hingegen nicht, ermöglicht es allerdings deutlich mehr Exploits zu analysieren. Der Umstieg auf Python 2 führt andererseits zu einem zusätzlichen Problem. Ab 2020 wird es keinen Support mehr für diese Pythonversion geben, sodass eine Migration zu Python 3 unabdingbar ist [77]. Ferner ergab die Recherche nach einer geeigneten Methode mithilfe von Python 3 ASTs für die in Python 2 verfassten Exploits zu erzeugen keine positiven Ergebnisse. Jedoch kann eine automatisierte Portierung, der in Python 2 verfassten Exploits zu Python 3, eine mögliche Lösung sein. Das Programm *2to3* ist dafür entwickelt worden diese Portierung vorzunehmen [74].

Damit zum Separieren der Exploits zwischen beiden Pythonversionen unterschieden werden kann, wird, wie in Abschnitt 5.2.4 erläutert, geprüft, ob im Quellcode ein Python3-Interpreter angegeben ist. Es hat sich gezeigt, dass dies keine optimale Lösung zur Unterscheidung der Versionen ist. Dies stellt indes nach einiger Recherche die einzige Möglichkeit dar. „Jenkins CVE-2016-0792 Deserialization Remote Exploit“ [65] ist ein Beispiel dafür, dass ein Python-Interpreter ohne Versionsnummer angegeben wurde, der Exploit hingegen in Python 3 verfasst ist. Dieser wird von parso als Exploit in Python 2 interpretiert, führt zu einer Behandlung als Quarantäne und kann nicht weiter analysiert werden.

Überdies stellte sich heraus, dass einige Exploits, welche in Python 2 verfasst wurden, während der Separierung dennoch nicht geparst werden konnten, da sie diverse Syntaxfehler aufwiesen. Dies wird als Fehler abgefangen und die Exploits werden als Quarantäne behandelt.

Die funktionale Anforderung FA07 sagt aus, dass bestehende Events in MISP mit den analysierten Ergebnissen der Exploits angereichert werden sollen. Nachdem verschiedene Exploits analysiert wurden, stellte sich heraus, dass eine Anreicherung nicht praktikabel anwendbar ist. Zum Auffinden von Events bedarf es einer Suche nach Attributen. Sofern diese Suche durchgeführt wird, muss entschieden werden, ob die gefundenen Events zu dem analysierten Exploit zugehörig sind. Müssen alle analysierten Ergebnisse in dem Event als Attribut enthalten sein, werden keine Events gefunden, da die Ergebnisse zu einem Exploit sehr spezifisch sind. Reicht das Auffinden von einem Ergebnis als ein At-

tribut in einem Event aus, führe dies zu einem *false positive*. Zwar wurde das Attribut korrekt gefunden, der Exploit kann allerdings nicht mit dem Event in Verbindung gebracht werden, da nach zu wenigen Merkmalen gefiltert wurde. Daher wird zu jedem analysierten Exploit ein neues Event angelegt. Ein Exploit kann dennoch mit anderen Events in Verbindung gebracht werden, da MISP eine Korrelation der Attribute vornimmt.

Bei der Entwicklung des MISPIntegrator-Moduls sind diverse Fehler in der REST-Schnittstelle zum Taggen von Events aufgefallen. Tabelle 5.1 zeigt mit welchen Taxonomien ein neues Event versehen werden soll. Der Ablauf zum Taggen sieht zunächst vor, dass die jeweiligen Taxonomien und Tags aktiviert werden müssen, bevor sie genutzt werden können. Eine Aktivierung erfolgt über die jeweilige ID einer Taxonomie bzw. eines Tags. Ferner werden in MISP die Taxonomien über einen Namespace - z.B. *europol-event* - identifiziert und bündeln eine Sammlung von Tags. Daher sind die in der allgemeinen Konfiguration der Analysesoftware definierten Taxonomien gleichzeitig die Bezeichner der Tags. Zur Ermittlung der Taxonomie-ID müssen alle in MISP vorhandenen Taxonomien mithilfe der Methode *get_taxonomies_list()* abgerufen werden. Anschließend findet ein Abgleich mit den jeweils zu verwendenden Taxonomien und der Liste aller statt. Allerdings wird nur ein Teil aller Taxonomien aus MISP abgerufen. Dadurch ist es nicht möglich neue Events mit den Taxonomien bzw. Tags *CERT-XLM:intrusion-attempts="exploit-known-vuln"* und *CERT-XLM:intrusion-attempts="new-attack-signature"* zu versehen, da sie nicht in der Liste enthalten sind. Dieser Fehler wurde auf GitHub dokumentiert [18].

Ein weiterer Fehler führt dazu, dass ein spezifischer Tag einer Taxonomie nicht aktiviert werden kann. Dies liegt daran, dass die Methode *get_tags_list()* nur aktivierte Tags ermittelt. Zur Aktivierung und Nutzung eines Tags ist allerdings die jeweilige ID notwendig. Diese kann nicht ermittelt werden, da die erwähnte Methode nicht korrekt arbeitet. Die einzige mögliche Lösung ist daher die Aktivierung aller Tags einer Taxonomie durch die vorliegende Taxonomie-ID. Auch dieser Fehler wurde auf GitHub dokumentiert [17].

Nachdem erste Analyseergebnisse vorlagen entstand der Eindruck, dass die bereits vorhandenen Metadaten aus Tabelle 4.2 mit weiteren Informationen, wie z.B. einer CVE-Nummer, erweitert werden könnten. Dadurch wäre ein separates Extrahieren dieser und anderer Informationen aus einem Exploit nicht mehr notwendig. Allerdings ist die Erweiterung um ein Feld für CVE-Nummern nicht durchführbar [16]. Auf die Frage zur

Erweiterung der Metadaten mit zusätzlichen Informationen, ist der Mitarbeiter von Offensive Security in dem GitHub Issue nicht eingegangen.

6.5 Anwendungsfall

Im Folgenden wird ein Anwendungsfall für die aus der Exploit-Analyse gewonnenen Ergebnisse geschildert. Über die ermittelten CVE-Nummern aus den Exploits aus dem Jahr 2018 werden die Schwachstellen und Advisories aus dem Advisory-System (ADV-System) des DFN-CERT abgerufen. Es soll anhand der darin gespeicherten CVSS-Werte überprüft werden, ob die Bewertungen von Schwachstellen und Advisories zutreffend sind. Dafür wird zunächst das Common Vulnerability Scoring System vorgestellt und anschließend das ADV-System. Im Anschluss wird der Anwendungsfall detaillierter vorgestellt und schließt mit einem Fazit.

Common Vulnerability Scoring System

Das Common Vulnerability Scoring System (CVSS) dient der standardisierten Bewertung von Schwachstellen in Firmware sowie Soft- und Hardware. Version 3.1 ist zur Zeit die aktuellste Version des Bewertungssystems. Hier wird allerdings Version 2.0 betrachtet, da das ADV-System die Version 3.1 bisher nicht unterstützt. CVSS setzt sich aus drei Metriken zusammen. Jede Metrik besitzt weitere Untermetriken die verschiedene Werte annehmen können. Jeder Wert besitzt nochmals einen Faktor, damit zu jeder der drei Metriken ein Score ermittelt werden kann. Diese Scores stellen Zahlenwerte dar, welche eine Schwachstelle bewertbar machen. Im folgenden werden die Metriken und Scores kurz vorgestellt. [19, 20]

Die *Base Metric* beschreibt die konstanten Eigenschaften einer Schwachstelle und ist immer gegeben. Der *Access Vector (AV)* beschreibt beispielsweise wie eine Schwachstelle ausgenutzt wird. Sie kann die Werte *Local (L)*, *Adjacent Network (A)* oder *Network (N)* annehmen. Local besitzt den Faktor 0,395, Adjacent Network den Faktor 0,646 und Network den Faktor 1,0. Somit ist ersichtlich in welcher Form ein Angreifender Zugriff auf ein System haben muss, um eine Schwachstelle auszunutzen. Weitere Metriken innerhalb der Base Metric sind *Access Complexity (AC)*, *Authentication (AU)*, *Confidentiality Impact (C)*, *Integrity Impact (I)* und *Availabilty Impact (A)*. Aus diesen Untermetriken werden zunächst der *Impact* und der *Exploitability Score* gebildet. Diese sind die Grundlage zur

Berechnung des Base Scores. Hierdurch wird eine Schwachstelle mit einem Wert in ihrer Schwere beschrieben und wird so mit anderen Bewertungen vergleichbar.

Die *Temporal Metric* stellt die zweite Metrik dar. Sie stellt eine über die Zeit ändernde Bedrohung durch die Schwachstelle dar. Da sich die Werte mit der Zeit ändern können und nicht immer vorliegen, ist die Angabe optional. Sie steht im Mittelpunkt des Anwendungsfalls und wird detaillierter beschrieben.

Tabelle 6.4 zeigt die möglichen Werte, die die *Exploitability (E)* Metrik annehmen kann. Sie bewertet die Verfügbarkeit eines Exploits als ein Programm oder Proof of Concept zur Ausnutzung einer Schwachstelle. Die Metrik *Remediation Level (RL)* wird zur Bewertung von möglichen Lösung zur Behebung einer Schwachstelle angewendet. Die existierenden Werte sind in Tabelle 6.5 aufgeführt. Die dritte Untermetrik innerhalb der Temporal Metric ist die *Report Confidence (RC)*. Sie gibt den Grad der Vertrauens- und Glaubwürdigkeit der Existenz einer Schwachstelle an. Ihre möglichen Werte sind in Tabelle 6.6 aufgelistet. Zur Berechnung des Temporal Scores werden die Faktoren der drei Untermetriken sowie der Base Score verwendet.

Bezeichnung	**Beschreibung**	**Faktor**
Unproven (U)	Es ist kein Exploit vorhanden, oder ein Exploit ist nur theoretisch anwendbar.	0,85
Proof of Concept (PoC)	Es existiert ein PoC der für die meisten Systeme nicht geeignet ist. Der PoC ist zudem nicht funktionsfähig oder bedingt einer Anpassung.	0,9
Functional (F)	Ein funktionsfähiger Exploit ist vorhanden und ist auf den meisten Systemen, auf denen die jeweilige Schwachstelle existiert, anwendbar.	0,95
High (H)	Die Schwachstelle ist durch ein vorhandenes funktionsfähiges Schadprogramm (Virus oder Wurm) in jeder Situation ausnutzbar.	1,00
Not Defined (ND)	Dieser Wert wird zur Berechnung des Temporal Scores nicht berücksichtigt.	1,00

Tabelle 6.4
Temporal Metric: Exploitability (E)

Bezeichnung	Beschreibung	Faktor
Official Fix (OF)	Die Schwachstelle wurde durch ein offizielles Update des Herstellers geschlossen.	0,87
Temporary Fix (TF)	Ein offizielles vom Hersteller bereitgestelltes temporäres Update ist vorhanden. Dies stellt allerdings nur einen Workaround oder ähnliches dar, welches die Schwachstelle schließt, aber nicht die optimale Lösung darstellt.	0,90
Workaround (W)	Eine nicht offizielle Lösung zur Schließung oder Abschwächung der Schwachstelle ist vorhanden.	0,95
Unavailable (U)	Es existiert keine Lösung zur Schließung der Schwachstelle oder es ist unmöglich eine bereitzustellen.	1,00
Not Defined (ND)	Dieser Wert wird zur Berechnung des Temporal Scores nicht berücksichtigt.	1,00

Tabelle 6.5
Temporal Metric: Remediation Level (RL)

Bezeichnung	Beschreibung	Faktor
Unconfirmed (UC)	Es existiert eine unbestätigte oder mehrere sich widersprechende Quellen die auf die Schwachstelle hinweisen. Das Vertrauen der Existenz der Schwachstelle ist demnach gering.	0,90
Uncorroborated (UR)	Mehrere nicht offizielle Quellen, wie z.B. Sicherheitsforscher*innen bestätigen die Existenz der Schwachstelle. Diese können hingegen uneindeutig sein oder sich widersprechen.	0,95
Confirmed (C)	Die Existenz der Schwachstelle wurde vom Hersteller bestätigt. Es wird zudem bestätigt, wenn bspw. ein funktionsfähiger Exploit oder ein Proof of Concept vorhanden ist.	1,00
Not Defined (ND)	Dieser Wert wird zur Berechnung des Temporal Scores nicht berücksichtigt.	1,00

Tabelle 6.6
Temporal Metric: Report Confidence (RC)

Die *Environmental Metric* gibt an, wie sich eine Schwachstelle auf die IT-Umgebung auswirken kann. Diese ist ebenfalls optional anzugeben. Sie besteht aus den drei Untermetriken *Collateral Damage Potential (CDP)*, *Target Distribution (TD)* sowie *Security Requirements*. Letztere ist nochmals in die drei Metriken *Confidentiality Requirement (CR)*, *Integrity Requirement (IR)* und *Availability Requirement (AR)* unterteilt. In die Berechnung des Environmental Scores fließen zudem der Base Score, diverse Werte der Base Metric und die der Temporal Metric ein.

Advisory-System

Mithilfe des ADV-Systems werden Schwachstellen für Hard- und Software beschrieben und bewertet. Zudem werden Advisories (Empfehlungen) verfasst, die Anweisungen zum Schließen von Sicherheitslücken geben, von denen die Kunden des DFN-CERTs betroffen sind. Innerhalb eines Advisories können mehrere Schwachstellen zusammengefasst beschreiben sein. Informationen zu den Schwachstellen kommen aus unterschiedlichen Quellen wie z.B. aus der National Vulnerability Database des NIST. Diese enthalten bereits eine Bewertung bzgl. der Base Metric und den berechneten Base Score. Eine Bewertung hinsichtlich der Temporal Metric wird manuell für jede Schwachstelle durchgeführt. Die Bewertung der Temporal Metric eines Advisories ergibt sich sodann aus den Bewertungen der verknüpften Schwachstellen. Wird eine Technologie von einem der Kunden des DFN-CERTs nicht unterstützt, wird hingegen keine Bewertung vorgenommen. Des Weiteren wird die Berechnung der Temporal Metric einer Schwachstelle bzw. Advisories lediglich nach ihrer direkten Betrachtung vorgenommen. Etwaige Aktualisierungen einer Schwachstelle werden somit nicht berücksichtigt bzw. stellen eine Ausnahme bei sehr bekannten und viel ausgenutzten Schwachstellen dar. Dadurch können sie nach einiger Zeit eine nicht mehr aktuelle Bewertung aufweisen.

Ziel dieses Anwendungsfalls soll die Erkennung von Inkonsistenzen bezüglich der Temporal Metric sein und die Identifizierung weiterer Auffälligkeiten zwischen den Bewertungen von Schwachstellen und Advisories. Eine Validierung der Temporal Metric in Abhängigkeit eines vorhandenen Exploits aus der ExploitDB zu einer Schwachstelle wird an dieser Stelle nicht vorgenommen. Dies bedarf einer Bewertung der Exploits nach einer Art Bewertungsschema hinsichtlich seiner Verfügbarkeit. Dadurch kann die Exploitability einer Schwachstelle oder eines Advisories verifiziert und ggf. angepasst werden. Auf der einen Seite ist ein solches Bewertungsschema nicht bekannt und auf der anderen

Seite müsste jeder Exploit individuell betrachtet und verstanden werden. Dennoch können die hier betrachteten Exploits aus der ExploitDB in die Untermetriken Proof of Concept, Functional oder High der Temporal Metric eingestuft werden. Ein Exploit aus der ExploitDB ist mindestens als ein PoC beschrieben und kann höchstens in einem funktionsfähigem Schadprogramm vorhanden sein. Einen Exploit als Unproven zu kategorisieren wäre falsch, da die Existenz eines Exploits nicht mehr theoretischer Natur ist. Weiterhin ist aus dem ADV-System nicht immer ersichtlich, ob bei der Bewertung einer Schwachstelle oder Advisories das Wissen über die Existenz eines Exploits einbezogen wurde.

In diesem Anwendungsfall wurden die Python-Exploits aus dem Jahr 2018 betrachtet aus denen CVE-Nummern extrahiert werden konnten. Es besitzen 116 Exploits mindestens eine CVE-Nummer. Insgesamt liegen 136 CVE-Nummern aus den Jahren 2013-2018 vor. Der Datenbestand des ADV-Systems ist vom 20.08.2019. Von diesen 136 CVE-Nummern besitzen 37 einen Base Score in dem ADV-System. Hiervon kommen 3 CVE-Nummern doppelt vor. 95 Nummern besitzen keinen Base Score und keine weiteren Bewertungen innerhalb des Systems. Davon treten 2 Nummern doppelt auf. Die übrigen 4 CVE-Nummern konnten in dem System nicht gefunden werden was anhand der dafür abgefragten Qullen zu erklären ist. Zudem kommt von diesen Nummern eine doppelt vor. Demnach konnten für 127 CVE-Nummern Informationen aus dem ADV-System abgefragt werden. Doppelt auftretende CVE-Nummern sind ein Hinweis dafür, dass zu einer Schwachstelle mehrere Exploits zur Verfügung stehen. Daher dürfen diese in der Betrachtung nicht ausgeblendet werden. Daher wurden 34 Schwachstellen betrachtet die innerhalb von 37 Exploits gefunden wurden und einen Base Score im ADV-System aufweisen.

Für den Vergleich der Bewertungen zwischen den Schwachstellen und den Advisories, wurden der Base Score, Temporal Score sowie die Untermetriken Exploitability, Remidiation Level und Report Confidence aus dem System abgerufen. Die dort gespeicherten Schwachstellen sind u.a. über die CVE-Nummer identifizierbar. Zwischen den Schwachstellen und den Advisories besteht eine n:m-Beziehung, sodass über die Identifikationsnummer der Schwachstelle alle mit ihr verknüpften Advisories ermittelt werden können. Zudem kann über das Veröffentlichkeits- und Erstellungsdatum der Schwachstellen und Advisories die zeitliche Differenz zu dem Veröffentlichkeitsdatum der jeweiligen Exploits berechnet werden. Dadurch kann deutlich gemacht werden, ob Schwachstellen und die damit verknüpften Advisories vor oder nach der Veröffentlichung eines Exploits offiziell publiziert und in das ADV-System eingetragen worden sind. Des Weiteren wird analy-

siert, ob mehrere CVE-Nummern aus einem Exploit in den Ergebnissen vorzufinden sind. Somit kann ermittelt werden, ob unterschiedliche Advisories mit den CVE-Nummern eines Exploits verknüpft sind.

Ergebnisse

Zu den 34 Schwachstellen wurden insgesamt 54 eindeutige Advisories ermittelt. Drei Schwachstellen weisen keine Verknüpfung zu Advisories auf. Zwei davon besitzen hingegen keine Bewertung hinsichtlich der Untermetriken der Temporal Metric. Der Temporal Score sollte laut der CVSS-Spezifikation folglich identisch mit dem Base Score sein [19]. Dies trifft indes zu.

Mit Ausnahme der Schwachstelle, die die Nummer CVE-2018-11529 trägt, wurden alle weiteren Schwachstellen vor der Veröffentlichung der jeweiligen Exploits veröffentlicht und in das ADV-System eingetragen. Diese einzige Schwachstelle wurde 6 Tage nach Veröffentlichung des Exploits offiziell veröffentlicht und am selben Tag in dem System notiert. Zudem wird die Schwachstelle in genau einem Advisory verwendet, welches 14 Tage nach Veröffentlichung des Exploits in das System erfasst und veröffentlicht wurde. Base und Temporal Score sowie die Exploitability, das Remediation Level und die Report Confidence sind jeweils konsistent zueinander.

Zusätzlich gibt es 12 Advisories die mit 9 hier betrachteten Schwachstellen verknüpft sind. Die Advisories wurden nach der Veröffentlichung des jeweiligen Exploits erstellt und veröffentlicht. Dies kann z.B. daran liegen, dass eine Schwachstelle bereits behoben wurde, dies allerdings nicht ausreichend war, sodass eine neue Schwachstelle diese Behebung umgeht. Der vorhandene Exploit zu der geschlossenen Schwachstelle sollte die neue Schwachstelle hingegen nicht ausnutzen können. Demnach kann nicht ausgesagt werden, dass es problematisch sei, wenn ein Advisory zeitlich nach der Veröffentlichung eines assoziierten Exploits erstellt wurde.

Tabelle 6.8 zeigt die Schwachstellen aus dem ADV-System, deren CVE-Nummern aus den Exploits aus dem Jahr 2018 extrahiert werden konnten und als Unproven eingestuft wurden. Diese Kategorisierung ist insofern nicht korrekt, da ein Exploit zur Ausnutzung der Schwachstelle vorhanden und nicht mehr theoretisch ist. Weiterhin fällt auf, dass die Schwachstellen, mit Ausnahme der Schwachstelle mit der Nummer CVE-2018-0296, einen geringeren Base Score aufweisen, als jener der in der National Vulnerability Database

(NVD) zu finden ist. Dies kann darauf zurückzuführen sein, dass die Informationen aus einer anderen Quelle als der NVD stammen.

Neben den vorhandenen Python-Exploits für die Schwachstellen CVE-2017-12635 und CVE-2017-12636, gibt es bereits ein Metasploit Modul zur praktischen Ausnutzung dieser [62]. Demnach wäre die Exploitability mindestens auf Functional zu setzen. Zudem sind beide Schwachstellen u.a. mit dem gleichen Advisory verknüpft, welches ausschließlich mit diesen Schwachstellen assoziiert ist. Dadurch kommt die identische Einstufung der Untermetriken der Temporal Metric zustande. Allerdings besitzt das Advisory einen Base Score von 6,5 und einen Temporal Score von 4,8. Die beiden Schwachstellen besitzen hingegen ganz andere Scores. Tabelle 6.7 zeigt diese Unterschiede. Das Advisory, welches ausschließlich mit den beiden Schwachstellen verknüpft ist bekommt hier die Bezeichnung „Advisory“. Außerdem sind die Exploit IDs angegeben, mit denen die Schwachstellen ausgenutzt werden können.

Wie der Unterschied hinsichtlich der Bewertungen zustande gekommen ist, kann nicht geklärt werden. Dennoch gilt hier die gleiche vorherige Argumentation, sodass eine Einstufung als Unproven nicht korrekt ist. Ein Grund für diese Einstufung kann die zu Beginn erwähnte Tatsache sein, dass die Schwachstellen im ADV-System im Nachhinein in der Regel nicht mehr aktualisiert werden. Die beiden Python-Exploits zu der Schwachstelle CVE-2017-12635 wurden knapp 5 bzw. 7 Monate nach Veröffentlichung der Schwachstelle veröffentlicht. Darunter befindet sich zudem der Exploit für die Schwachstelle CVE-2017-12636.

Bezeichnung	**Base Score**	**Temporal Score**	**E**	**RL**	**RC**	**Exploit ID(s)**
CVE-2017-12635	6,0	4,4	U	OF	C	44498, 44913
CVE-2017-12636	7,9	5,8	U	OF	C	44913
Advisory	6,5	4,8	U	OF	C	44498, 44913

Tabelle 6.7
Gegenüberstellung der CVSS-Bewertungen zweier Schwachstellen und einem mit ihnen assoziierten Advisory

Des Weiteren scheint es eine Inkonsistenz bei der Schwachstelle mit der Nummer CVE-2018-0296 zu geben, unabhängig davon, dass es einen Python-Exploit gibt. Es gibt genau ein Advisory, welches mit dieser Schwachstelle assoziiert ist, allerdings als Functional kategorisiert wurde. In den Kommentaren des Advisory ist zu sehen, dass einige Wochen nach dessen Veröffentlichung ein Exploit veröffentlicht wurde. Aus diesem Grund wurde die Exploitability angepasst. Allerdings nicht die der Schwachstelle.

Ähnliches gilt für die Schwachstelle mit der Nummer CVE-2018-6789. Es existieren 2 Python-Exploits für die Schwachstelle und sie wird mit genau 2 Advisories assoziiert. Beide Advisories sind zudem nur mit dieser Schwachstelle verknüpft. Ein Advisory besitzt die identische Bewertung wie die Schwachstelle. Das andere Advisory hingegen wurde als PoC eingestuft. Wie dies zustande kam, ist aus dem ADV-System nicht ersichtlich.

Eine weitere Inkonsistenz weist die Schwachstelle CVE-2018-7600 auf. Diese ist als Functional eingestuft. Ein damit verknüpftes Advisory, welches ausschließlich mit dieser Schwachstelle assoziiert ist, besitzt hingegen die Einstufung High. Aus den Kommentaren des Advisory und der Schwachstelle ist dennoch zu entnehmen, dass ein Exploit vorhanden ist und aktiv ausgenutzt wird. Die Schwachstelle und das Advisory wurden am 29.03.2018 veröffentlicht und in das System eingetragen. Laut Advisory ist seit dem 11.04.2018 der Exploit bekannt. Ein Kommentar über die Existenz des Exploits wurde der Schwachstelle hingegen erst am 26.04.2018 beigefügt. Insgesamt zeigt dieses Beispiel, dass die Aktualisierung zum einen nicht konsistent durchgeführt wurde und zum anderen zeitlich sehr verzögert ist.

Dieser Anwendungsfall zeigt exemplarisch auf, dass das ADV-System diverse Inkonsistenzen aufweist. Zudem gibt es verschiedene Bewertungen von Schwachstellen und Advisories die nicht korrekt sind. Dies kann darauf zurückzuführen sein, dass lediglich einige wenige Daten in dem System nach ihrer ersten Veröffentlichung aktualisiert werden. Dennoch ist zu erkennen, dass die Schwachstellen i.d.R. zeitlich vor der Veröffentlichung eines geeigneten Exploits in das System eingetragen werden.

CVE-Nummer	Base Score (ADV)	Base Score (NVD)	Temporal Score	E	RL	RC	Exploit ID(s)	Anzahl Advisories
CVE-2017-12635	6,0	10,0 [56]	4,4	U	OF	C	44498, 44913	1
CVE-2017-12636	7,9	9,0 [57]	5,8	U	OF	C	44913	2
CVE-2018-0296	7,8	5,0 [58]	5,8	U	OF	C	44956	1
CVE-2018-6789	6,8	7,5 [59]	5,0	U	OF	C	44571, 45671	2
CVE-2018-7254	4,3	6,8 [60]	3,2	U	OF	C	44154	1

Tabelle 6.8
Auswahl von Schwachstellen aus dem ADV-System die als Unproven eingestuft wurden aus dem Jahr 2018

7 Zusammenfassung

7.1 Fazit

Der Kern dieser Arbeit war die Vorstellung eines Konzeptes zur Entwicklung eines Prototyps zur automatisierten Analyse von öffentlich zugänglichen Exploits. Zudem wurde die technische Realisierung geschildert und die Ergebnisse der Exploit-Analyse in einer umfangreichen Evaluation ausgewertet. Diese Ergebnisse wurden als Cyber Threat Intelligence angesehen.

Zur Einordnung des Begriffes Cyber Threat Intelligence, wurde dieser zunächst genauer erörtert. Es wurden die Definitionen nach Chismon und Ruks [4] und nach Johnson u. a. [27] vorgestellt. Anschließend wurden diese miteinander verglichen sowie die Vorteile und Herausforderungen für den Einsatz von CTI herausgestellt. Danach wurde ein CTI-Lifecycle vorgestellt. Somit war es möglich die entwickelte Analysesoftware darin zu verorten. Es wurde dadurch deutlich, dass es sich um eine Software handelt, die für das Konsumieren sowie Produzieren von CTI einzusetzen ist.

Zur Repräsentation von Bedrohungsinformationen gibt es diverse Formate, von denen einige etablierte vorgestellt wurden. Diese sind zusätzlich für einen einheitlichen Austausch zwischen verschiedene Parteien notwendig und sind in die Collection-Phase des CTI-Lifecycles zu verorten. Das MISP Core Format stand dabei im besonderen Fokus, da die gewonnenen Analyseergebnisse nach MISP exportiert werden sollten und diese Plattform eine steigende Beliebtheit für den Austausch von CTI aufweist. Anschließend wurden etablierte Protokolle für den Austausch von CTI diskutiert. Eine genauere Betrachtung von MISP und der ExploitDB - als Quelle der in dieser Arbeit ausgewerteten Exploits - folgten im Anschluss. Eine weitere Zielsetzung war die Benennung von Zielgruppen von CTI. Darum wurde im weiteren Verlauf der Arbeit eine Einordnung der gewonnenen Ergebnisse aus der Exploit-Analyse zu diesen Gruppen vorgenommen.

Aus der im ersten Teil der Arbeit definierten Zielsetzung sowie den vorgestellten Grundlagen wurden einige Forschungsfragen und Hypothesen abgeleitet. Dabei stand die Frage im Vordergrund, welche Hinweise zur Erkennung von Vorfällen aus den Exploits identifiziert werden können. Zusätzliche Fragen und Hypothesen zielten auf die Gewinnung weiterer interessanter Erkenntnisse über existierende Exploits. Im Anschluss wurde eine Eingliederung der Analysesoftware in den CTI-Lifecycle vorgenommen. Daran anknüpfend wurden für eine erfolgreiche Umsetzung eines Prototyps diverse funktionale und nicht-funktionale Anforderungen vorgestellt. Aus diesen Anforderungen wurde das Design der Software entwickelt und aufgezeigt. Die Aufgaben der einzelnen Komponenten wurden anschließend ausführlich beschrieben. Die Komponente ExploitDB stellt hierbei den Kern der Software dar. In ihr wurde das Abrufen der Exploits aus der ExploitDB, die Separierung der Exploits sowie die Analyse von Kommentaren und Code realisiert. In dem darauf folgenden Abschnitt wurde die technische Umsetzung erläutert. Darin wurden zum einen die verwendeten technischen Hilfsmittel wie Bibliotheken oder die genutzte Datenbank erläutert. Zum anderen wurde die Umsetzung der einzelnen Komponenten verdeutlicht aus denen u.a. die Arbeitsweise der Kommentar- und Codeanalyse hervorgeht.

In der Evaluation wurden zunächst die aufgestellten Forschungsfragen und Hypothesen u.a. mit erstellten Diagrammen beantwortet. Darauf folgend wurden weitere Diagramme vorgestellt, wodurch diverse Trends bei der Entwicklung von Python-Exploits erkennbar wurden. Zuletzt wurden die Hindernisse die während der Entwicklung auftraten diskutiert. Die Arbeit schließt mit der Vorstellung eines Anwendungsfalls ab. Darin wurden die Bewertungen der Schwachstellen und Advisories aus dem ADV-System des DFN-CERT analysiert. Dies wurde durch die Ermittlung der CVE-Nummern aus den analysierten Python-Exploits aus dem Jahr 2018 realisiert.

Mit dieser Arbeit wurde erfolgreich gezeigt, dass eine neue Quelle für CTI erschlossen wurde. Einige der gewonnenen Hinweise aus der Exploit-Analyse wurden in die Kategorien *precursor* und *indicator* eingeteilt und können zur Erkennung von Sicherheitsvorfällen dienlich sein. Der Anteil dieser Hinweise beträgt 30% von allen ermittelten Hinweisen. Allerdings wurde festgestellt, dass dieser Wert nicht bewertet werden kann, da zunächst analysiert werden muss, ob die Hinweise in der Praxis zur Erkennung von Vorfällen hilfreich sind. Weiterhin wurden die gewonnenen Hinweise in die beiden Kategorien Technical sowie Tactical Threat Intelligence eingeordnet. Durch den exemplarischen Anwendungsfall konnten diverse aus den Exploits extrahierte CVE-Nummern verwendet werden. Dadurch wurde ersichtlich, dass das ADV-System des DFN-CERT diverse Inkonsistenzen

hinsichtlich der Bewertungen von Advisories aufweist. Zusätzlich wurde nochmals deutlich, dass es ein Problem darstellt, wenn die Daten in dem System nicht kontinuierlich aktualisiert werden.

Durch die Kategorisierung der gewonnenen Ergebnisse und der Vorstellung der Zielgruppen von CTI konnten alle Zielgruppen als Konsumenten dieser Informationen eingestuft werden. Ferner sollte ein Zusammenhang zwischen der Anzahl veröffentlichter Schwachstellen und der Anzahl der bereitgestellten Exploits hergestellt werden. Es wurde deutlich, dass zum einen nicht zu jedem Exploit eine Schwachstelle dokumentiert sein muss und zum anderen, dass nicht jeder Exploit einen Verweis auf eine veröffentlichte Schwachstelle enthalten muss. Aus der Gegenüberstellung dieser Werte zwischen den Jahren 2009 und 2018 wurde deutlich, dass diese eine hohe Differenz pro Jahr aufweisen können. Infolgedessen kann nicht gesagt werden, dass eine Steigerung der Anzahl von Schwachstellen eine Steigerung der Anzahl von Exploits nach sich zieht.

Ein weiterer wichtiger Aspekt war der zeitliche Zusammenhang zwischen der Veröffentlichung von Schwachstellen und Exploits. Es wurde aufgezeigt, dass eine automatisierte Umsetzung zur Analyse eines zeitlichen Zusammenhangs derzeit nicht vorhanden ist, da die ExploitDB kein Datum der Veröffentlichung von Schwachstellen enthält. Ein manueller Vergleich für einen zeitlichen Zusammenhang ist durch eine Analyse der Events in der MISP-Oberfläche jedoch möglich. Eine kleinere manuelle Analyse des zeitlichen Zusammenhangs wurde zudem in dem Anwendungsfall durchgeführt und geschildert. Dieser machte deutlich, dass in der Regel nach der Veröffentlichung einer Schwachstelle ein dazugehöriger Exploit bereitgesetllt wurde. Die zeitliche Differenz schwankte jedoch zwischen wenigen Tagen und einigen Monaten oder Jahren.

Zuletzt konnten die Forschungsfragen zur Popularität von Plattformen und eingesetzter Programmiersprachen bzgl. der Exploits beantwortet werden. Allerdings konnte die Hypothese zur Popularität von Plattformen und der damit einhergehenden höheren Verwundbarkeit durch existierende Exploits nicht eindeutig beantwortet werden. Das lag in erster Linie an der ungenauen Datenlage und fehlenden offiziellen Statistiken zu diesem Aspekt. Hinsichtlich der Popularität von Programmiersprachen und jener die zur Entwicklung von Exploits verwendet werden, wurde festgestellt, dass diesbezüglich kein Zusammenhang besteht. Jedoch wurden Tendenzen dahingehend erkannt, dass je beliebter oder geeigneter eine Sprache ist, diese zudem für die Entwicklung von Exploits eingesetzt wird.

Weitere statistische Auswertungen zeigten, dass zwischen den Jahren 2009 und 2018 deutlich mehr Python-Exploits veröffentlicht wurden, als in den davor liegenden 21 Jahren zusammen. Zudem war ersichtlich, dass der Anteil der Python-Exploits in diesem Zeitraum zumeist der zweit höchste war, im Gegensatz zu allen anderen Exploits die in einer anderen Programmiersprache verfasst wurden. Exploits die in Textform verfasst sind, treten am häufigsten auf. Dies zeigt die hohe Popularität der Programmiersprache bei der Entwicklung von Exploits auf.

Eine weitere Besonderheit stellte sich bei der statistischen Analyse der Typen von Sicherheitslücken heraus. Die Auswertung der Beschreibungen hinsichtlich dieser Typen verdeutlichte, dass deutlich mehr Exploits eines Typs vorhanden sind, als auf der offiziellen ExploitDB-Webseite angegeben sind.

7.2 Ausblick

Die Arbeit hat einen ersten Prototypen zur Analyse von Exploits und Extrahierung von Hinweisen zur Erkennung von Sicherheitsvorfällen aufgezeigt. Dies ist nach aktuellem Wissensstand die erste wissenschaftliche Auseinandersetzung damit. Es zeigte sich zudem, dass der Prototyp diverse Schwächen und Erweiterungspotential aufweist die eine Weiterentwicklung notwendig macht. Im Fokus steht besonders der Anteil der extrahierten Indikatoren die zur Erkennung von Vorfällen notwendig sind. Es war zu sehen, dass viele ermittelte Hinweise nicht in die Kategorien von Bedrohungsinformationen eingeordnet werden konnten. Sie zeigen interessante Informationen zu Exploits auf, sind jedoch nicht verwertbar für die Erkennung von Vorfällen.

Die folgende Auflistung stellt einen kleinen Einblick für Korrekturen und Erweiterungen oder der weiteren Forschung zur Verfügung. Die Reihenfolge der Punkte deutet nicht auf ihre Wichtigkeit hin.

- Herausarbeiten weiterer Indikatoren die aus den Exploits extrahiert werden können.
- Parsen von Exploits die in anderen Programmiersprachen verfasst sind.
- In Anlehnung an FF05 kann eine zusätzliche Funktionalität entwickelt werden, die den zeitlichen Zusammenhang zwischen der Veröffentlichung neuer Schwachstellen und Exploits mithilfe von CVE-Nummern automatisiert herstellt. Dies sieht vor,

dass die Schwachstelleninformationen bspw. aus der NVD abgerufen werden, sodass das Datum der CVEs mit denen der Exploits verglichen werden kann.

- Es wäre die Fragen zu klären, ob die gewonnenen Indikatoren in der Praxis hilfreich sind. Dazu müsste jeweils von den Zielgruppen bewertet werden, ob sie von Nutzen sind.

- Wie bereits in Abschnitt 6.4 erörtert, endet der Support für Python 2 Anfang 2020, sodass eine Migration zu Python 3 mittelfristig unabdingbar ist. Daher muss eine Lösung zum Parsen von Python 2 Code in Python 3 ermittelt werden. Eine Möglichkeit könnte sein, die Python-Exploits zunächst automatisiert nach Python 3 zu portieren, um anschließend die Analyse durchführen zu können. *2to3* bietet diese Möglichkeit der Portierung [74].

- Die MISP-Attributes sind den Hinweisen nicht zuzuordnen, sodass nicht ersichtlich ist, um welche Information es sich genau handelt. Stattdessen kann bspw. in dem Kommentarfeld eines Attributes, die Bezeichnung des Hinweises hinterlegt werden.

- Bisher erhalten alle Events in MISP die gleiche Klassifizierung in Form einer Taxonomie. Wünschenswert wäre es, wenn während der Analyse erkannt wird, welche Taxonomie am sinnvollsten für den jeweiligen Exploit erscheint. Dies könnte z.B. in Abhängigkeit des Typs der Sicherheitslücke (Buffer Overflow, etc.) durchgeführt werden.

- Die Entwicklung einer Art Bewertungsschema für Exploits hinsichtlich ihrer Verfügbarkeit. Es scheint sinnvoll, eine Möglichkeit zu besitzen, die die Temporal Metric einer Schwachstelle im Bezug auf ihrer Exploitability in Abhängigkeit eines vorhandenen Exploits verifiziert.

- Während der Betrachtung des Anwendungsfalls, ist der Autor dieser Arbeit auf ein Mapping von CVE-Nummern auf die Exploits aus der ExploitDB gestoßen [49]. Daraus kann eine Komponente zur Ergänzung von CVE-Nummern zu den Exploits entwickelt werden.

Eine Weiterentwicklung und Optimierung der Analysesoftware von Exploits begegnet der zügig wachsenden Verwendung und dem Interesse von Cyber Threat Intelligence. Aus diesem Grund wird sie öffentlich zugänglich gemacht, sodass Forschende oder Unternehmen darauf zugreifen können [78].

Literaturverzeichnis

[1] Bhaskar, Rahul: State and Local Law Enforcement is not Ready for a CYBER KATRINA. In: *Communications of the ACM* 49 (2006), Februar, Nr. 2, S. 81–83. – URL https://cacm.acm.org/magazines/2006/2/6014-state-and-local-law-enforcement-is-not-ready-for-a-cyber-katrina/. – Zugriffsdatum: 30.01.2020

[2] Bundesamt für Sicherheit in der Informationstechnik (BSI): Leitfaden „IT-Forensik“ / Bundesamt für Sicherheit in der Informationstechnik (BSI). URL https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Dienstleistungen/IT-Forensik/forensik_node.html, März 2011 (1.0.1). – Forschungsbericht. Zugriffsdatum: 30.01.2020

[3] Bundesamt für Sicherheit in der Informationstechnik (BSI): Die Lage der IT-Sicherheit in Deutschland 2018 / Bundesamt für Sicherheit in der Informationstechnik (BSI). URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2018.pdf, September 2018. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[4] Chismon, David ; Ruks, Martyn: Threat I]ntelligence: Collecting, Analysing, Evaluating / MWR InfoSecurity, CERT-UK, CPNI. URL https://www.mwrinfosecurity.com/assets/Whitepapers/Threat-Intelligence-Whitepaper.pdf, 2015. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[5] Cichonski, Paul ; Millar, Tom ; Grance, Tim ; Scarfone, Karen: Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology / National Institute of Standards and Technology (NIST). National Institute of Standards and Technology, August 2012. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[6] Cisco: Cisco 2018 Annual Cybersecurity Report / Cisco Systems, Inc. URL https://www.cisco.com/c/dam/m/digital/elq-cmcglobal/witb/acr2018/acr2018final.pdf, Februar 2018. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[7] Danyliw, R.: The Incident Object Description Exchange Format Version 2 / RFC Editor. RFC Editor, November 2016 (7970). – RFC. – URL https://tools.ietf.org/html/rfc7970. Zugriffsdatum: 30.01.2020. – ISSN 2070-1721

[8] Darley, Trey ; Kirillov, Ivan: *STIX^{TM} Version 2.0. Part 5: STIX Patterning.* Juli 2017. – URL http://docs.oasis-open.org/cti/stix/v2.0/cs01/part5-stix-patterning/stix-v2.0-cs01-part5-stix-patterning.pdf. – Zugriffsdatum: 30.01.2020

[9] Data Is Beautiful: *Most Popular Programming Languages 1965 - 2019.* – URL https://www.invidio.us/watch?v=Og847HVwRSI. – Zugriffsdatum: 30.01.2020

[10] Dempsey, Tim ; Rosenquist, Matt (Hrsg.): *Navigating The Digital Age.* Caxton Business & Legal inc, Oktober 2015. – URL https://www.securityroundtable.org/wp-content/uploads/2015/09/Cybersecurity-9780996498203-no_marks.pdf. – Zugriffsdatum: 30.01.2020. – ISBN 978-0-9964982-0-3

[11] Dulaunoy, Alexandre ; Iklody, Andras: MISP core format / IETF Secretariat. URL https://www.ietf.org/internet-drafts/draft-dulaunoy-misp-core-format-07.txt, February 2019 (draft-dulaunoy-misp-core-format-07). – Internet-Draft. Zugriffsdatum: 30.01.2020

[12] Endler, Matthias: *GitHub - mre/awesome-static-analysis: Static analysis tools for all programming languages.* – URL https://github.com/mre/awesome-static-analysis#python. – Zugriffsdatum: 30.01.2020

[13] European Network and information Security Agency (ENISA): Good Practice Guide for Incident Management / European Union Agency for Network and Information Security (ENISA). URL https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management, 2010. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[14] European Union Agency for Cybersecurity (ENISA): *Is Software More Vulnerable Today?*. – URL https://www.enisa.europa.eu/publications/info-notes/is-software-more-vulnerable-today. – Zugriffsdatum: 30.01.2020

[15] European Union Agency For Network and Information Security (ENISA): Exploring the opportunities and limitations of current Threat Intelligence Platforms / European Union Agency for Network and Information Security (ENISA). URL https://www.enisa.europa.eu/publications/exploring-the-opportunities-and-limitations-of-current-threat-intelligence-platforms, Dezember 2017. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[16] fabrei: *expand csv file · Issue #124 · offensive-security/exploitdb · GitHub*. – URL https://github.com/offensive-security/exploitdb/issues/124. – Zugriffsdatum: 30.01.2020

[17] fabrei: *How to really get all Tags? · Issue #343 · MISP/PyMISP · GitHub*. – URL https://github.com/MISP/PyMISP/issues/343. – Zugriffsdatum: 30.01.2020

[18] fabrei: *Taxonomy list is not complete · Issue #342 · MISP/PyMISP · GitHub*. – URL https://github.com/MISP/PyMISP/issues/342. – Zugriffsdatum: 30.01.2020

[19] FIRST.org, Inc.: *CVSS v2 Complete Documentation*. – URL https://www.first.org/cvss/v2/guide. – Zugriffsdatum: 30.01.2020

[20] FIRST.org, Inc.: *CVSS v3.1 Specification Document*. – URL https://www.first.org/cvss/v3.1/specification-document. – Zugriffsdatum: 30.01.2020

[21] FIRST.org, Inc.: *FIRST - Improving Security Together*. – URL https://www.first.org/. – Zugriffsdatum: 30.01.2020

[22] Flory, Teri: Digital Forensics in Law Enforcement: A Needs Based Analysis of Indiana Agencies. In: *Journal of Digital Forensics, Security and Law* 11 (2016), Nr. 1. – URL https://commons.erau.edu/cgi/viewcontent.cgi?article=1374&context=jdfsl. – Zugriffsdatum: 30.01.2020

[23] Foord, Michael: *Mock - Mocking and Testing Library — Mock 3.0.5 documentation.* – URL https://mock.readthedocs.io/en/latest/. – Zugriffsdatum: 30.01.2020

[24] Halter, Dave: *parso - A Python Parser.* – URL https://parso.readthedocs.io/en/latest/. – Zugriffsdatum: 30.01.2020

[25] Hunter, J. D.: Matplotlib: A 2D graphics environment. In: *Computing in Science & Engineering* 9 (2007), Nr. 3, S. 90–95. – Zugriffsdatum: 30.01.2020

[26] IstiZada: *List of EMEA Countries.* – URL http://istizada.com/list-of-emea-countries/. – Zugriffsdatum: 30.01.2020

[27] Johnson, Chris ; Badger, Lee ; Waltermire, David ; Snyder, Julie ; Skorupka, Clem: NIST Special Publication 800-150 - Guide to Cyber Threat Information Sharing / National Institute of Standards and Technology (NIST). URL http://dx.doi.org/10.6028/NIST.SP.800-150, Oktober 2016. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[28] Jordan, Bret ; Piazza, Rich ; Wunder, John: *STIX™ Version 2.0. Part 2: STIX Objects.* Juli 2017. – URL https://docs.oasis-open.org/cti/stix/v2.0/cs01/part2-stix-objects/stix-v2.0-cs01-part2-stix-objects.pdf. – Zugriffsdatum: 30.01.2020

[29] Kampanakis, Panos: Security Automation and Threat Information-Sharing Options. In: *IEEE Security & Privacy* 12 (2014), sep, Nr. 5, S. 42–51. – URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6924671. – Zugriffsdatum: 30.01.2020

[30] Kent, Karen ; Chevalier, Suzanne ; Grance, Tim ; Dang, Hung: Guide to Integrating Forensic Techniques into Incident Response / National Institute of Standards and Technology (NIST). URL https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-86.pdf, August 2006. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[31] Leach, Paul J. ; Mealling, Michael ; Salz, Rich: A Universally Unique IDentifier (UUID) URN Namespace / RFC Editor. RFC Editor, July 2005 (4122). – RFC. – URL https://tools.ietf.org/html/rfc4122. Zugriffsdatum: 30.01.2020. – ISSN 2070-1721

[32] McKinney, Wes: Data Structures for Statistical Computing in Python. In: Walt, Stéfan van der (Hrsg.) ; Millman, Jarrod (Hrsg.): *Proceedings of the 9th Python in Science Conference*, 2010, S. 51 – 56. – Zugriffsdatum: 30.01.2020

[33] MISP: *Extending MISP with Python modules.* – URL https://www.circl.lu/assets/files/misp-training/switch2016/2-misp-modules.pdf. – Zugriffsdatum: 30.01.2020

[34] MISP: *MISP CVE Module.* – URL https://github.com/MISP/misp-modules/blob/master/misp_modules/modules/expansion/cve.py. – Zugriffsdatum: 30.01.2020

[35] MISP: *MISP Default Feeds.* – URL https://www.misp-project.org/feeds/. – Zugriffsdatum: 30.01.2020

[36] MISP: *MISP Galaxy.* – URL https://github.com/MISP/misp-galaxy. – Zugriffsdatum: 30.01.2020

[37] MISP: *MISP Galaxy Android.* – URL https://github.com/MISP/misp-galaxy/blob/master/clusters/android.json. – Zugriffsdatum: 30.01.2020

[38] MISP: *MISP Malware Classification Taxonomy.* – URL https://github.com/MISP/misp-taxonomies/blob/master/malware_classification/machinetag.json. – Zugriffsdatum: 30.01.2020

[39] MISP: *MISP Modules.* – URL https://github.com/MISP/misp-modules. – Zugriffsdatum: 30.01.2020

[40] MISP: *MISP Objects.* – URL https://www.misp-project.org/objects.html. – Zugriffsdatum: 30.01.2020

[41] MISP: *MISP Objects Exploit-PoC.* – URL https://www.misp-project.org/objects.html#_exploit_poc. – Zugriffsdatum: 30.01.2020

[42] MISP: *MISP Objects Relationships.* – URL https://github.com/MISP/misp-objects/blob/master/relationships/definition.json. – Zugriffsdatum: 30.01.2020

[43] MISP: *MISP Taxonomies.* – URL https://github.com/MISP/misp-taxonomies. – Zugriffsdatum: 30.01.2020

[44] MISP: *MISP taxonomies and classification as machine tags - CERT-XLM:intrusion-attempts="exploit-known-vuln"*. – URL https://www.misp-project.org/taxonomies.html#_cert_xlmintrusion_attemptsexploit_known_vuln. – Zugriffsdatum: 30.01.2020

[45] MISP: *MISP taxonomies and classification as machine tags - CERT-XLM:intrusion-attempts="new-attack-signature"*. – URL https://www.misp-project.org/taxonomies.html#_cert_xlmintrusion_attemptsnew_attack_signature. – Zugriffsdatum: 30.01.2020

[46] MISP: *MISP taxonomies and classification as machine tags - cyber-threat-framework:Engagement="exploit-vulnerabilities"*. – URL https://www.misp-project.org/taxonomies.html#_cyber_threat_frameworkengagementexploit_vulnerabilities. – Zugriffsdatum: 30.01.2020

[47] MISP: *MISP taxonomies and classification as machine tags - enisa:nefarious-activity-abuse="exploits-exploit-kits"*. – URL https://www.misp-project.org/taxonomies.html#_enisanefarious_activity_abuseexploits_exploit_kits. – Zugriffsdatum: 30.01.2020

[48] MISP: *MISP taxonomies and classification as machine tags - europol-event:exploit-tool-exhausting-resources*. – URL https://www.misp-project.org/taxonomies.html#_europol_eventexploit_tool_exhausting_resources. – Zugriffsdatum: 30.01.2020

[49] MITRE Corporation: *CVE - CVE Reference Map for Source EXPLOIT-DB*. – URL https://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html. – Zugriffsdatum: 30.01.2020

[50] MongoDB, Inc: *The most popular database for modern apps | MongoDB*. – URL https://www.mongodb.com/. – Zugriffsdatum: 30.01.2020

[51] MongoDB, Inc: *PyMongo 3.8.0 Documentation*. – URL https://api.mongodb.com/python/3.8.0/. – Zugriffsdatum: 30.01.2020

[52] Moriarty, K.: Real-time Inter-network Defense (RID) / RFC Editor. RFC Editor, April 2012 (6545). – RFC. – URL https://tools.ietf.org/html/rfc6545. Zugriffsdatum: 30.01.2020. – ISSN 2070-1721

[53] Moskowitz, Robert ; Karrenberg, Daniel ; Rekhter, Yakov ; Lear, Eliot ; Groot, Geert J. de: *Address Allocation for Private Internets*. RFC 1918. Februar 1996 (Request for Comments). – URL https://tools.ietf.org/html/rfc1918. – Zugriffsdatum: 30.01.2020

[54] National Institute of Standards and Technology: *NVD - CVE-2017-0022*. – URL https://nvd.nist.gov/vuln/detail/CVE-2017-0022. – Zugriffsdatum: 30.01.2020

[55] National Institute of Standards and Technology (NIST): *National Vulnerability Database*. – URL https://nvd.nist.gov/. – Zugriffsdatum: 30.01.2020

[56] National Institute of Standards and Technology (NIST): *NVD - CVE-2017-12635*. – URL https://nvd.nist.gov/vuln/detail/CVE-2017-12635. – Zugriffsdatum: 30.01.2020

[57] National Institute of Standards and Technology (NIST): *NVD - CVE-2017-12636*. – URL https://nvd.nist.gov/vuln/detail/CVE-2017-12636. – Zugriffsdatum: 30.01.2020

[58] National Institute of Standards and Technology (NIST): *NVD - CVE-2018-0296*. – URL https://nvd.nist.gov/vuln/detail/CVE-2018-0296. – Zugriffsdatum: 30.01.2020

[59] National Institute of Standards and Technology (NIST): *NVD - CVE-2018-6789*. – URL https://nvd.nist.gov/vuln/detail/CVE-2018-6789. – Zugriffsdatum: 30.01.2020

[60] National Institute of Standards and Technology (NIST): *NVD - CVE-2018-7254*. – URL https://nvd.nist.gov/vuln/detail/CVE-2018-7254. – Zugriffsdatum: 30.01.2020

[61] National Institute of Standards and Technology (NIST): *NVD - Statistics*. – URL https://nvd.nist.gov/vuln/search/statistics?form_type=Advanced&results_type=statistics&search_type=all&pub_start_date=01%2F01%2F2009&pub_end_date=12%2F31%2F2018. – Zugriffsdatum: 30.01.2020

[62] Offensive Security: *Apache CouchDB - Arbitrary Command Execution (Metasploit) - Linux remote Exploit.* – URL https://www.exploit-db.com/exploits/45019. – Zugriffsdatum: 30.01.2020

[63] Offensive Security: *Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers.* – URL https://www.exploit-db.com/. – Zugriffsdatum: 30.01.2020

[64] Offensive Security: *Exploit Database Submission Guidelines.* – URL https://www.exploit-db.com/submit. – Zugriffsdatum: 30.01.2020

[65] Offensive Security: *exploitdb/42394.py at master · offensive-security/exploitdb · GitHub.* – URL https://github.com/offensive-security/exploitdb/blob/master/exploits/java/remote/42394.py. – Zugriffsdatum: 30.01.2020

[66] Offensive Security: *exploitdb/46848.py at master · offensive-security/exploitdb · GitHub.* – URL https://github.com/offensive-security/exploitdb/tree/master/exploits/windows/dos/46848.py. – Zugriffsdatum: 30.01.2020

[67] Offensive Security: *exploitdb/files_exploits.csv at master · offensive-security/exploitdb · GitHub.* – URL https://github.com/offensive-security/exploitdb/blob/master/files_exploits.csv. – Zugriffsdatum: 30.01.2020

[68] Offensive Security: *The official Exploit Database repository.* – URL https://github.com/offensive-security/exploitdb. – Zugriffsdatum: 30.01.2020

[69] Office of the Director of National Intelligence (ODNI): *National Intelligence Strategy of the United States of America.* – URL https://www.dni.gov/files/ODNI/documents/National_Intelligence_Strategy_2019.pdf. – Zugriffsdatum: 30.01.2020

[70] Offizielle STIX Dokumentation: *Sighting SRO describing an Indicator being seen.* – URL https://oasis-open.github.io/cti-documentation/img/NewSTIXdiagram2.PNG. – Zugriffsdatum: 30.01.2020

[71] Offizielle TAXII Dokumentation: *TAXII Diagramm.* – URL https://oasis-open.github.io/cti-documentation/img/taxii_diagram2.png. – Zugriffsdatum: 30.01.2020

[72] Pace, Chris ; Barysevich, Andrei ; Gundert, Levi ; Liska, Allan ; McDaniel, Maggie ; Wetzel, John ; Ahlberg, Dr. C.: *The Threat Intelligence Handbook*. CyberEdge Group, LLC, 2018. – URL https://cyber-edge.com/wp-content/uploads/2018/11/Recorded-Future-eBook.pdf. – Zugriffsdatum: 30.01.2020. – ISBN 978-0-9990354-7-4

[73] Ponemon Institute LLC: Third Annual Study on Exchanging Cyber Threat Intelligence: There Has to Be a Better Way / Ponemon Institute LLC. URL https://www.infoblox.com/wp-content/uploads/infoblox-white-paper-ponemon-infoblox-2018-final-report.pdf, Januar 2018. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[74] Python Software Foundation: *25.4. 2to3 - Automated Python 2 to 3 code translation — Python v3.1.5 documentation*. – URL https://docs.python.org/3.1/library/2to3.html. – Zugriffsdatum: 30.01.2020

[75] Python Software Foundation: *32.2. ast — Abstract Syntax Trees — Python 2.7.16 documentation*. – URL https://docs.python.org/2/library/ast.html. – Zugriffsdatum: 30.01.2020

[76] Python Software Foundation: *7.2. re — Regular expression operations — Python 2.7.16 documentation*. – URL https://docs.python.org/2/library/re.html. – Zugriffsdatum: 30.01.2020

[77] Python Software Foundation: *Sunsetting Python 2 | Python.org*. – URL https://www.python.org/doc/sunset-python-2/. – Zugriffsdatum: 30.01.2020

[78] Reiber, Fabian: *Exploit Analyzer*. – URL https://github.com/fabianHAW/exploit_analyzer. – Zugriffsdatum: 30.01.2020

[79] Reiber, Fabian: Überblick diverser Plattformen zum Verarbeiten von Sicherheitsvorfällen und wie sie entsprechende Informationen untereinander austauschen können. Juni 2017. – Forschungsbericht

[80] Sauerwein, Clemens ; Sillaber, Christian ; Mussmann, Andrea ; Breu, Ruth: Threat Intelligence Sharing Platforms: An Exploratory Study of Software Vendors and Research Perspectives. In: *Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik (WI 2017)*, URL https://pdfs.semanticscholar.

org/0b9b/00a2dbf6ae467395fac917e0f7b73cc3e7aa.pdf, Februar 2017, S. 837–851. – Zugriffsdatum: 30.01.2020

[81] SecurityTrails Team: *Exploring Google Hacking Techniques.* – URL https://securitytrails.com/blog/google-hacking-techniques. – Zugriffsdatum: 30.01.2020

[82] Serketzis, Nikolaos ; Katos, Vasilios ; Ilioudis, Christos ; Baltatzis, Dimitrios ; Pangalos, Georgios: Improving Forensic Triage Efficiency through Cyber Threat Intelligence. In: *Future Internet* 11 (2019), jul, Nr. 7, S. 162. – Zugriffsdatum: 30.01.2020

[83] Shackleford, Dave: Cyber Threat Intelligence Uses, Successes and Failures: The SANS 2017 CTI Survey / SANS Institute. URL https://www.sans.org/reading-room/whitepapers/threats/cyber-threat-intelligence-uses-successes-failures-2017-cti-survey-37677, März 2017. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[84] Shackleford, Dave: CTI in Security Operations: SANS 2018 Cyber Threat Intelligence Survey / SANS Institute. URL https://www.sans.org/reading-room/whitepapers/threats/cti-security-operations-2018-cyber-threat-intelligence-survey-38285, Februar 2018. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[85] Sheridan, Kelly: *How Cybercriminals Exploit Simple Human Mistakes.* – URL https://www.darkreading.com/threat-intelligence/how-cybercriminals-exploit-simple-human-mistakes/d/d-id/1335847. – Zugriffsdatum: 30.01.2020

[86] Skopik, Florian ; Settanni, Giuseppe ; Fiedler, Roman: A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing / Digital Safety and Security Department, Austrian Institute of Technology, Donau-City-Straße 1, 1220 Vienna, Austria. URL http://www.flosko.at/ait/2016_cs.pdf, April 2016. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[87] Slatman, Herman: *GitHub - hslatman/awesome-threat-intelligence: A curated list of Awesome Threat Intelligence resources.* – URL https://github.com/hslatman/awesome-threat-intelligence. – Zugriffsdatum: 30.01.2020

[88] Stack Overflow Network: *Stack Overflow Insights - Developer Hiring, Marketing, and User Research.* – URL https://insights.stackoverflow.com/survey/. – Zugriffsdatum: 30.01.2020

[89] StatCounter: *Desktop, Mobile & Tablet Operating System Market Share Worldwide | StatCounter Global Stats.* – URL https://gs.statcounter.com/os-market-share/desktop-mobile-tablet/worldwide/#monthly-200901-201812. – Zugriffsdatum: 30.01.2020

[90] The Open Group: *Risk Taxonomy.* The Open Group, Januar 2009. – URL https://pubs.opengroup.org/onlinepubs/9699919899/toc.pdf. – Zugriffsdatum: 30.01.2020. – ISBN 1-931624-77-1

[91] Torres, Alissa: Building a World-Class Security Operations Center: A Roadmap / SANS Institute. URL https://sibertor.com/wp-content/uploads/2016/07/building-world-class-security-operations-center-roadmap-35907.pdf, Mai 2015. – Forschungsbericht. Zugriffsdatum: 30.01.2020

[92] Tounsi, Wiem ; Rais, Helmi: A survey on technical threat intelligence in the age of sophisticated cyber attacks. In: *Computers & Security* 72 (2018), Januar, S. 212–233. – Zugriffsdatum: 30.01.2020

[93] Trammell, B.: Transport of Real-time Inter-network Defense (RID) Messages over HTTP/TLS / RFC Editor. RFC Editor, April 2012 (6546). – RFC. – URL https://tools.ietf.org/html/rfc6546. Zugriffsdatum: 30.01.2020. – ISSN 2070-1721

[94] Trier, Michael ; Thiel, Sebastian: *GitPython Documentation.* – URL https://gitpython.readthedocs.io/en/stable/. – Zugriffsdatum: 30.01.2020

[95] Vinot, Raphaël: *PyMISP Documentation.* – URL https://buildmedia.readthedocs.org/media/pdf/pymisp/latest/pymisp.pdf. – Zugriffsdatum: 30.01.2020

[96] Wagner, Cynthia ; Dulaunoy, Alexandre ; Wagener, Gérard ; Iklody, Andras: 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 - WISCS'16*, ACM Press, 2016. – Zugriffsdatum: 30.01.2020

[97] Wunder, John ; Davidson, Mark ; Jordan, Bret: *TAXII™ Version 2.0 - Committee Specification 01*. Juli 2017. – URL https://docs.oasis-open.org/cti/taxii/v2.0/cs01/taxii-v2.0-cs01.pdf. – Zugriffsdatum: 30.01.2020

[98] Zimmerman, Carson: *Ten Strategies of a World-Class Cybersecurity Operations Center*. MITRE Corporation, 2014. – URL https://www.mitre.org/sites/default/files/publications/pr-13-1028-mitre-10-strategies-cyber-ops-center.pdf. – Zugriffsdatum: 30.01.2020. – ISBN 978-0-692-24310-7

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Masterarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Exemplarische Erzeugung von Cyber Threat Intelligence (CTI) aus einer Exploit-Analyse und deren Integration in die Malware Information Sharing Platform (MISP)

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original