HAW HAMBURG

Bachelorarbeit

Lars Schneider

Analyse von Angriffen auf Honeypots zur Erzeugung von Warnmeldungen und Indicators of Compromise

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Lars Schneider

Analyse von Angriffen auf Honeypots zur Erzeugung von Warnmeldungen und Indicators of Compromise

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Technische 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. Olaf Zukunft

Eingereicht am: 19. September 2019

Lars Schneider

Thema der Arbeit

Analyse von Angriffen auf Honeypots zur Erzeugung von Warnmeldungen und Indicators of Compromise

Stichworte

Indicators of Compromise, IoC, Honeypot, Threat Intelligence

Kurzzusammenfassung

In dieser Arbeit wird ein Prozess entwickelt und eingesetzt, mit dem Ereignisse, die an Honeypotsensoren passieren, auf erfolgte Angriffe untersucht werden. Außerdem wird eine Automation anhand des entwickelten Prozesses programmiert. Mit dieser werden für eingehende Angriffe Warnmeldungen und Indicators of Compromise erzeugt.

Lars Schneider

Title of Thesis

Analysis of attacks on honeypots to generate alerts and Indicators of Compromise

Keywords

Indicators of Compromise, IoC, Honeypot, Threat Intelligence

Abstract

In this work, a process is developed and used to investigate events that happen on honeypot sensors for attacks. Also an automation based on the developed process will be programmed. With this automation warning messages and indicators of compromise will be generated for incoming attacks.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Bedrohungen von Netzwerken und deren Komponenten sind heutzutage allgegenwertig. Daher spielt die Netzwerksicherheit in Unternehmen eine immer größere und wichtigere Rolle, um Ressourcen und Informationen vor Dritten zu schützen. Dabei findet ein stetiges Wettrüsten statt. Auf der einen Seite sind Sicherheitsexperten und Entwickler bemüht, Lücken in Soft- und Hardware zu schließen, beziehungsweise Methoden und Möglichkeiten zu entwickeln, um sich gegen aktuelle Schadsoftware zu schützen. Auf der anderen Seite versuchen Hacker, neue Sicherheitslücken zu entdecken und neue Schadsoftware zu programmieren und zu verbreiten.

Schon im frühen 20. Jahrhundert hat der französische Forensiker Edmond Locard festgestellt: Every Contact leaves a trace.[17] Dieses Zitat, aus einem der wichtigsten Prinzipien der modernen Forensik, kann auch auf die Informationstechnik angewendet werden.

Jeder Kontakt mit einem Netzwerk oder System hinterlässt Spuren. Allerdings ist es bei der Flut von Informationen, die in einem Netzwerk eingehen, schwierig, diese zu filtern und zu analysieren, um gegebenenfalls Bedrohungen identifizieren zu können und sich dagegen zu schützen. Indicators of Compromise (IoC) bieten einerseits die Möglichkeit, identifizierte Bedrohungen formalisiert zu beschreiben und über vertrauensvolle Stellen mit anderen Unternehmen zu teilen. Andererseits können Netzwerke und Systeme anhand von IoC auf bereits vorhandene Bedrohungen überprüft werden.

In Verbindung mit einem Honeypot, können IoC neue Bedrohungen in einer geschützten Umgebung identifizieren und beschreiben, ohne dass das eigene Netzwerk gefährdet wird.

1.1 Ziel der Arbeit

Das Ziel der Arbeit ist es, Angriffe, die auf einem Honeypot erfolgen, zu identifizieren, zu analysieren und formal zu beschreiben. Der Honeypot soll dabei aufgezeichnete Daten an eine Instanz liefern, die die vom Honeypot gelieferten Informationen sammelt und strukturiert. Anschließend soll die Instanz die Datensammlung analysieren und mit ihr erfolgte Angriffe und Angriffsszenarien formalisiert durch IoC beschreiben. Die erzeugten IoC sollen als Warnmeldung an eine zentrale Stelle gesandt werden.

Die Instanz zur Sammlung und Analyse der vom Honeypot gelieferten Daten, sowie zur Erzeugung der IoC, soll im Rahmen dieser Arbeit konzipiert, implementiert und getestet werden und sich dabei an einem ebenfalls zu erstellenden Alarmierungsprozess halten. Der Alarmierungsprozess lehnt sich an einen bereits existierenden Prozess an, welcher im weiteren Verlauf der Arbeit vorgestellt wird.

1.2 Zielgruppe

Diese Arbeit richtet sich an alle Personen, die sich für Netzwerksicherheit interessieren. Im Besonderen richtet sich die Arbeit an Netzwerksicherheitsexperten und Netzwerkadministratoren, die Maßnahmen gegen Bedrohungen im eigenen Netzwerk treffen müssen oder das Netzwerk vorbeugend gegen potentielle Bedrohungen schützen wollen.

1.3 Aufbau der Arbeit

In Kapitel 2 werden grundlegende Themen erläutert, die zum Verständnis der weiteren Arbeit benötigt werden. Es wird beschrieben, was Honeypots sind, in welchen Varianten sie existieren und welche Vor- bzw. Nachteile ein Einsatz von Honeypots bietet. Im Anschluss daran, werden IoC als Objektformat zur Darstellung von sicherheitsrelevanten Informationen - Cyber Threat Intelligence (CTI) - vorgestellt und ihre Funktion und ihr Nutzen grundlegend geschildert. Abschließend wird in Kapitel 2 ein Alarmierungsprozess für Intrusion Detection Systems vorgestellt, anhand dessen sich der in dieser Bachelorarbeit zu entwickelnde Alarmierungsprozess orientieren soll.

Kapitel 3 beschäftigt sich mit der Konzipierung, der für das Ziel der Arbeit benötigten Komponenten. Es werden die eingesetzten Honeypotsysteme und die eingesetzten Standards vorgestellt. Außerdem wird, anhand des in Kapitel 2 vorgestellten Prozesses, ein Alarmierungsprozess konzipiert, mit dem eingehende Angriffe analysiert und Warnmeldungen erzeugt werden sollen.

Kapitel 4 beinhaltet die Implementation und Konfiguration der zuvor konzipierten Komponenten und des Alarmierungsprozesses. Des Weiteren wird der implementierte Prozess auf Funktionalität getestet.

Im 5. und letzten Kapitel werden die Erkenntnisse der Arbeit gesammelt und bewertet. Des Weiteren wird beurteilt, ob das Ziel der Arbeit erreicht wurde. Abschließend erfolgt ein Ausblick.

2 Grundlagen

Wie in Kapitel 1 beschrieben, werden in Kapitel 2 die Grundlagen zum Verständnis der weiteren Arbeit erläutert. Es werden die Einsatzziele, die Funktion und die Kategorisierung von Honeypots mit ihren Vor- und Nachteilen erklärt.

Anschließend werden in diesem Kapitel die verschiedenen Typen von IoC, mit ihrer Wichtigkeit für verschiedene Institutionen und Personen, erleutert.

Am Ende des Kapitels wird ein an der Universität von Kalifornien entwickelter Alarmierungsprozess vorgestellt, der entwickelt wurde, um eine Übersicht über sicherheitsrelevante Aktivitäten innerhalb eines Netzwerkes zu bekommen.

2.1 Honeypots

Ein Honeypot ist definitionsgemäß eine Computerressource, deren Wert darin liegt, aufgespürt, angegriffen und kompromittiert zu werden.[19, vgl. S.58]

Die Ziele des Einsatzes eines Honeypots variieren je nach Einsatzort und Zweck. Teilweise werden sie zu reinen Forschungszwecken eingesetzt, deren Ziel es ist, neue Angriffsmethoden und Schadsoftware der Hacker zu identifizieren und zu dokumentieren.

Unternehmen setzen Honeypots ein, um Angriffe auf ihr Netzwerk abzufangen. Dadurch erhalten sie die Möglichkeit, ihr Netzwerk frühzeitig gegen Angriffe zu schützen und Bedrohungen zu verringern.

2.1.1 Honeypot Varianten

Honeypots können in Forschungs- und Produktionshoneypots eingeteilt werden, die sich in ihren Einsatzzielen deutlich unterscheiden.

Produktionshoneypots helfen Unternehmen dabei, das Risiko eines erfolgreichen Angriffes auf ihr Netzwerk zu verringern und somit die Sicherheit des Netzes zu steigern. Sie melden unberechtigte Zugriffe, die auf dem jeweiligen Honeypot erfolgen. Da ein Honeypot keinen produktiven Nutzen in einem Netzwerk einnimmt, sind jegliche Zugriffe als unberechtigt und somit als Angriff zu werten. Durch Meldung der Zugriffe auf den Honeypot, kann das produktive System gegen die jeweilige Bedrohung geschützt werden.[19, vgl. S.61f]

Produktionshoneypots sind einfacher zu implementieren als Forschungshoneypots, da sie weniger Analysefähigkeiten benötigen und somit einen geringeren Funktionsumfang haben. Dadurch, dass sie mit geringer Funktionalität eingesetzt werden, stellen sie aber auch ein geringeres Risiko für das Unternehmen dar. Die eingeschränkte Funktionalität macht es Angreifern schwieriger, mit einem übernommenen Honeypot, Schaden im produktiven Netzwerk eines Unternehmens anzurichten. [19, vgl. S.62]

Forschungshoneypots werden speziell dazu eingesetzt, um Informationen über Angreifer und Angriffsmethoden zu sammeln. Die so gesammelten Daten können unter anderem dazu genutzt werden, Strategien und Methoden zur Verteidigung zu entwickeln, Sicherheitslücken zu schließen und Angreifer zu identifizieren. [19, vgl. S82f]

Forschungshoneypots werden häufig von Sicherheitsunternehmen eingesetzt, die die gesammelten Daten dazu nutzen, ihre Kunden entsprechend vor den auftretenden Bedrohungen zu warnen oder zu schützen. Auch staatliche Einrichtungen und Behörden betreiben diese Art von Honeypots.

2.1.2 Interaktionsgrad eines Honeypots

Abseits von der Kategorisierung nach dem Einsatzziel, können Honeypots nach ihrem Interaktionsgrad in drei Typen klassifiziert werden. High-, Medium- und Low-Interaction Honeypots.

Low-Interaction Honeypots sind im Prinzip nur Gerüste von Computersystemen oder Servern. Sie simulieren nach außen hin Dienste, auf die zugegriffen werden kann. Sie haben aber kein reales Betriebssystem, auf welches über die simulierten Dienste zugegriffen und

mit dem interagiert werden könnte. Wenn ein Angreifer, die von einem Low-Interaction Honeypot angebotenen Dienste angreift, wird er nach einem erfolgreichen Zugriff nur sehr begrenzt dazu in der Lage sein, weiter mit dem Honeypot zu agieren bzw. die Kontrolle darüber zu übernehmen. [4, vgl.]

Diese Art von Honeypot wird lediglich dazu eingesetzt, um festzustellen, ob Angriffe stattfinden. Dabei können Informationen, über z.B. die Quell und Ziel IP-Adresse, sowie über die Zeit und das Datum des Angriffes, gesammelt werden. Eine genauere Analyse des Angreifers und seiner Methoden, ist mit einem Low-Interaction Honeypot schwer umzusetzen.

High-Interaction Honeypots hingegen sind voll funktionsfähige Computersysteme mit realen Betriebssystemen. Entgegen den Low-Interaction Honeypots, haben Angreifer hier die Möglichkeit, nach dem Ausnutzen einer Schwachstelle, voll mit dem System zu interagieren und es im Zweifel zu übernehmen. Durch die erweiterten Interaktionsmöglichkeiten fallen mehr Daten an, wodurch erweiterte Analysemöglichkeiten geboten werden, um Informationen über Angreifer und deren Methoden zu identifizieren. [4, vgl.]

Medium-Interaction Honeypots bieten größere Interaktionsmöglichkeiten als Low-Interaction Honeypots, sind aber nicht so umfassend konfiguriert und ausgestattet, wie High-Interaction Honeypots.

Sie verfügen über kein komplettes reales Betriebssystem, mit dem Angreifer interagieren könnten. Allerdings sind die Dienste, die sie anbieten, ausgiebig konfiguriert. So können Angreifer beispielsweise, nach einem erfolgreichen Angriff auf einen vom Honeypot angebotenen Web-Dienst, voll mit diesem interagieren. Allerdings haben sie kaum Möglichkeiten, von dem Web-Dienst aus weiter in das Honeypotsystem vorzudringen. [19, vgl. S.94]

Tabelle 2.1: Vergleich der Interaktionsgrade [19, vgl. S91]

Interaktionsgrad	Aufwand	Analyse	Risiko
Low	Einfach	Limitiert	Gering
Medium	Unterschiedlich	Variabel	Medium
High	Hoch	Umfassend	Hoch

Die Tabelle 2.1 setzt die einzelnen Interaktionsgrade von Honeypots hinsichtlich ihres Aufwandes für die Installation, Konfiguration, Bereitstellung und Wartung, sowie den

Umfang der Informationssammlung und des jeweiligen Risiko Levels in Vergleich.

So hat der Einsatz eines Low-Interaction Honeypots, durch den geringen Funktionsumfang und die somit geringen Analysemöglichkeiten, einen sehr geringen Implementations- und Wartungsaufwand. Des Weiteren ist das Risiko, welches durch den Einsatz des Honeypots besteht und der Umfang der gesammelten Informationen über erfolgte Angriffe ebenfalls niedrig.

Die Einrichtung eines Medium-Interaction Honeypots kann sehr komplex werden. Angebotene Dienste sollen möglichst real und umfangreich in ihren Funktionen und ihren Interaktionsmöglichkeiten sein. Allerdings soll ein Angreifer, von den angebotenen Diensten aus, nicht weiter in das System des Honeypots vordringen können.
Es besteht eine Gradwanderung, zwischen einerseits der Einrichtung eines möglichst umfangreichen Dienstes, mit auf der anderen Seite geringem Risiko für das System, auf dem der Honeypot implementiert ist.
So kann der Aufwand eines Medium-Interaction Honeypots, je nach Komplexität des eingesetzten Dienstes, sehr komplex werden.

High-Interaction Honeypots nehmen ein, im Verhältnis zu den anderen Interaktionsgraden, großes Risiko in Kauf, dass ein Angreifer das Honeypot System übernehmen könnte, da sie über ein reales Betriebssystem verfügen, mit dem Angreifer interagieren können. Dafür haben sie aber, durch ihren Funktions- und Interaktionsumfang, auch die größten Analysemöglichkeiten unter allen verfügbaren Honeypot Varianten. Allerdings ist der Aufwand der Installation und des Betriebs eines solchen Honeypots auch entsprechend hoch.

2.1.3 Honeypot und Honeynet

Bei einer Verbindung von mehreren Honeypots zu einem Netzwerk, spricht man von einem Honeynet. Honeynets sind das Extrem der High-Interaction Honeypots. [19, vgl. S.99]

Durch die Bildung eines Honeynets, kann ein produktives Netzwerk komplett nachgebildet werden. Ein Honeynet bietet durch die Anzahl der eingesetzten Systeme eine größere Angriffsfläche, als ein einzelner Honeypot und hat somit eine verstärkte Präsenz, womit das Honeynet für Angreifer ein deutlich attraktiveres Ziel ist.

Durch den größeren Aufwand, den es benötigt, ein Honeynet einzurichten, werden sie hauptsächlich als Forschungshoneypots zur Analyse eingesetzt. Für Unternehmen, die sich mit Hilfe eines Honeypots gegen Angriffe schützen wollen, ist der Einsatz eines Honeynets meist zu zeit- und kostenaufwändig. [19, vgl. S237]

2.1.4 Platzierung von Honeypots

Honeypots können in unterschiedlichen Bereichen sinnvoll platziert werden. Die Platzierung eines Honeypots hängt dabei von den jeweiligen Einsatzzielen und der jeweils gegebenen Infrastruktur des Netzwerkes ab. Hauptsächlich werden Honeypots im internen Netzwerk, in der Demilitarisierte Zone (DMZ) oder im Internet platziert.

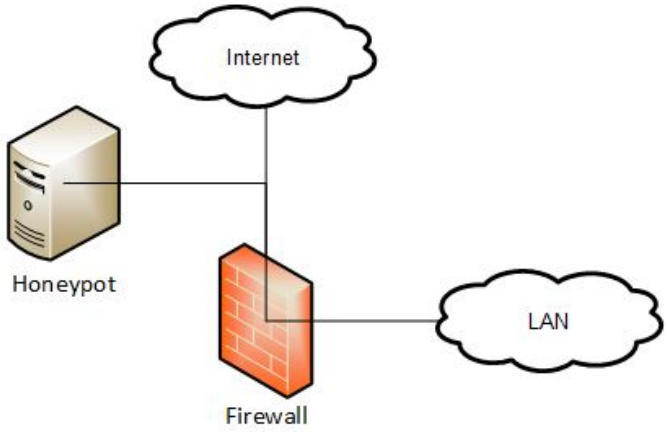

Abbildung 2.1: Platzierung im Internet

Externe, im Internet platzierte Honeypots (Abb. 2.1) werden durch ihre exponierte Stellung das Ziel vieler Angriffe. Daher ist es wichtig, die Honeypot Umgebung vom restlichen System ausreichend zu separieren und zu schützen, um eine Übernahme des kompletten Systems zu vermeiden. Durch Monitoring und Alarmierungsfunktionen kann, im Falle einer Kompromittierung, das System zeitnah ausgeschaltet werden, um zu vermeiden, dass es für die Zwecke des Angreifers eingesetzt wird. Durch die große Anzahl an Angriffen, die auf einen extern platzierten Honeypot erfolgen, ist eine dortige Platzierung besonders interessant für Forschungshoneypots. [16, vgl. S55f]

Für Unternehmen ist die externe Platzierung eines Honeypots meist nicht von besonders großem Nutzen. Es empfiehlt sich ihnen eher, Honeypots innerhalb der DMZ des Unternehmens zu platzieren (Abb. 2.2). Innerhalb der DMZ imitieren Honeypots produktive

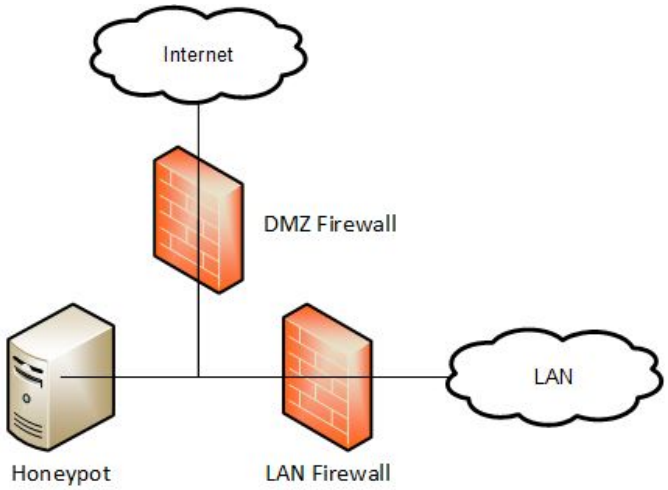

Abbildung 2.2: Platzierung in der DMZ

Systeme und lenken damit von diesen ab. Zusätzlich dienen Honeypots innerhalb der DMZ als Frühwarnsystem. Im Falle eines Angriffes können sowohl produktive Systeme der DMZ, als auch das interne Netzwerk frühzeitig geschützt werden.

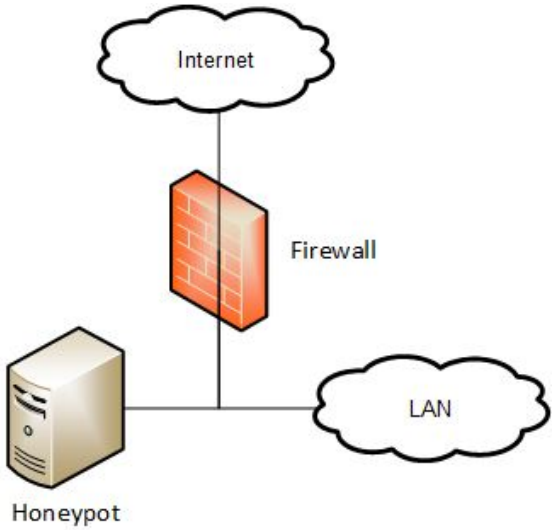

Abbildung 2.3: Platzierung im produktiven Netzwerk

So, wie bei einer Platzierung in der DMZ, imitieren Honeypots bei einer Platzierung im internen Netzwerk (Abb. 2.3) ein produktives System. Sie können damit auch als Frühwarnsystem fungieren, wobei sie dabei besonders gegen Angriffe aus dem internen Netzwerk warnen und schützen können.

2.1.5 Vor- und Nachteile des Einsatzes eines Honeypots

Der Einsatz von Honeypots bietet die Möglichkeit, durch Datensammlung und Analyse, neue Sicherheitsmechanismen gegen Malware und andere Bedrohungen zu entwickeln. Für Unternehmen kann der Einsatz von Honeypots die Sicherheit erhöhen, indem Honeypots als Frühwarnsystem von Angriffen fungieren und den Angreifer zusätzlich verlangsamen.

Auf der anderen Seite besteht bei einem Einsatz von Honeypots ein erhöhtes Risiko für das produktive System eines Unternehmens. Wird ein Honeypot kompromittiert und übernommen, so kann er dazu genutzt werden, andere Systeme des Netzwerkes zu kompromittieren. Wird ein übernommener Honeypot zum Angriff auf Dritte verwendet, kann dies rechtliche Konsequenzen für das Unternehmen des Honeypots nach sich ziehen. Um eine Übernahme von eingesetzten Honeypots zu verhindern, müssen diese regelmäßig gewartet und überwacht werden.

2.2 Indicators of Compromise

Indicators of Compromise sind Daten, die in einem Netzwerk oder auf einem Computersystem beobachtet werden und auf einen erfolgten Angriff oder eine Kompromittierung des jeweiligen Systems hindeuten können. [18, vgl.]

IoC können jeweils einzelne Datensätze, wie z.B. eine IP-Adresse beschreiben, oder aber aus einer Sammlung von Datensätzen bestehen, die unter anderem ein komplettes Angriffsmuster beschreiben können.

Typischerweise werden IoC durch einen der verfügbaren Standards dargestellt, die dazu entwickelt wurden, um CTI zu beschreiben und formal darzustellen. Zu diesen Standards gehören insbesondere OpenIoC, IODEF (Incident Object Description Exchange Format) und STIX (Structured Threat Information eXpression). [14, S.14]

Neben den Standards zur Beschreibung von CTI, wurde ein weiterer Standard entwickelt, der zur Verteilung der beschriebenen Informationen dient. Dieser Standard heißt TAXII (Trustet Automated Exchange of Intelligence Information) und wurde ursprünglich dazu entwickelt, in STIX dargestellte Informationen auszutauschen. Allerdings kann TAXII auch dazu genutzt werden, um Daten zu verteilen, die mit Formaten aus anderen Standards beschrieben wurden. [13, vgl.]

2.2.1 Funktion und Nutzen

Die Verwendung von IoC bietet mehrere Vorteile. Einerseits werden einzelne Bedrohungen auf eine standardisierte Weise beschrieben, wodurch sie für eine große Masse verständlich sind und somit ohne Probleme verbreitet und geteilt werden können. Andererseits können IoC auch automatisch durch vertrauenswürdige Quellen an Sicherheitssysteme verteilt werden, die dann wiederrum ihr eigenes Netzwerk nach denen, durch die IoC beschriebenen Bedrohungen, durchsuchen können. Des Weiteren bieten IoC die Möglichkeit, anhand beschriebener Signaturen oder Methoden, neue Sicherheitsfeatures zu entwickeln, um möglicherweise zukünftigen Zero-Day Angriffen vorzubeugen. [14, vgl. S.15]

Damit Unternehmen und andere Institutionen oder Personen von IoC profitieren können, müssen diese zunächst erzeugt werden und anschließend auf ein entsprechendes Sicherheitssystem verteilt werden, damit dieses in seinem Netzwerk, anhand der erzeugten IoC, nach Bedrohungen suchen kann.

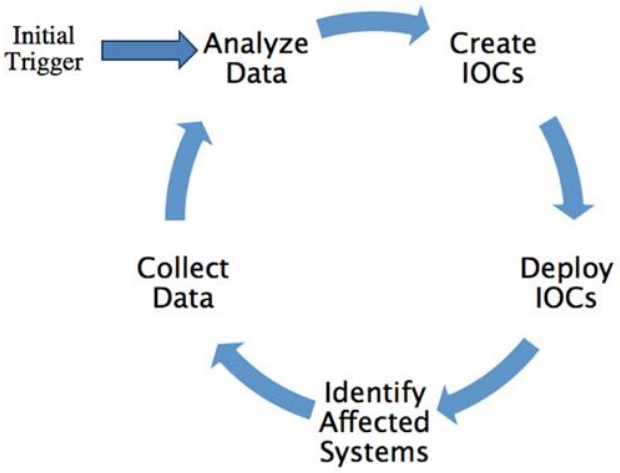

Abbildung 2.4: IoC Lebenszyklus [14, S.14]

Dabei folgen IoC einem Lebenszyklus, der in Abbildung 2.4 dargestellt wird.

Bevor ein IoC erzeugt werden kann, muss ein auslösendes Ereignis, wie z.B. die Kompromittierung eines Systems passieren.

Werden bei der Analyse des kompromittierten Systems Daten gefunden, die Rückschluss auf die Ursache oder die Quelle der Kompromittierung geben, kann anhand dieser Daten ein IoC erzeugt werden.

Der erzeugte IoC kann anschließend auf geeigneten Sicherheitssystemen installiert werden, um das System oder das gesamte Netzwerk nach den Daten zu durchsuchen, die auf eine erneute Kompromittierung oder einen Kompromittierungsversuch hindeuten.

Wird anhand des erzeugten IoC erneut ein System identifiziert, auf welchem die kompromittierenden Daten beobachtet werden, können die Daten auf diesem System wiederum analysiert werden, um den vorhandenen IoC mit neueren Daten zu aktualisieren oder zu erweitern oder um einen komplett neuen IoC zu erzeugen.

2.2.2 Arten und Klassifizierung

Es existieren verschiedenste Daten und Informationen, die mit Hilfe von Indicators of Compromise beschrieben werden können und wo IoC bei der Analyse von, und Schutz vor Netzwerkbedrohungen helfen können.

Um einen Überblick über die Arten von Indicators of Compromise zu erhalten und deren Wertigkeit zu bestimmen, hat David Bianco die in Abbildung 2.5 dargestellte Pyramid of Pain erstellt. Die Beschreibung der Pyramide bezieht sich auf einen von David Bianco verfassten Artikel aus dem Jahr 2013. [15]

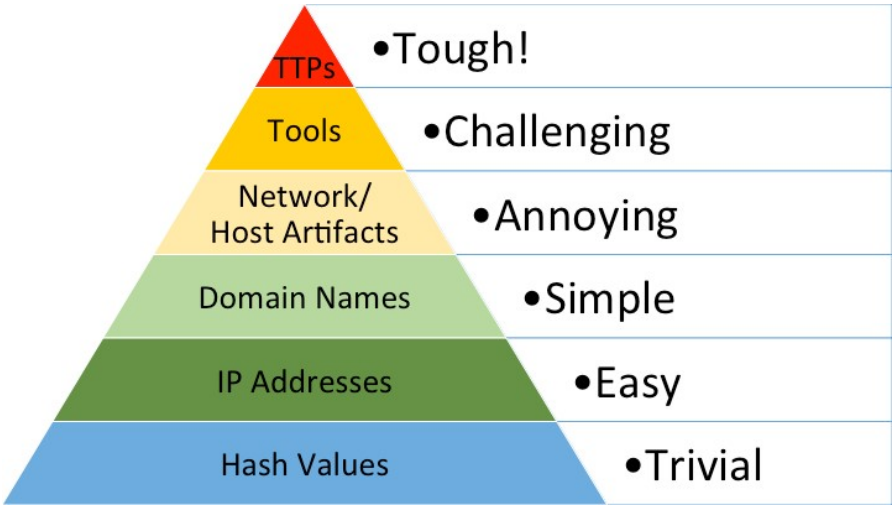

Abbildung 2.5: Pyramid of Pain [15]

In dieser Pyramide werden verschiedene Datenkategorien abgebildet, aus denen IoC erstellt werden können. Zusätzlich wird dargestellt, welchen Verbreitungsgrad und welchen

Wert diese Daten für den Kampf gegen Netzwerkbedrohungen haben, bzw. welcher Aufwand seitens der Hacker betrieben werden muss, um die jeweiligen Daten einer Kategorie in z.B. ihrer Malware zu erneuern oder anzupassen. Unten in der Pyramide werden die Daten dargestellt, die am meisten verbreitet sind und deren Entdeckung den Angreifern am wenigsten schadet. Höhere Kategorien sind immer weniger verbreitet und bedeuten bei Entdeckung ein immer höheres Ärgernis für die Angreifer.

Ganz unten in der Pyramide sind Hash Values zu finden. Es ist nicht schwierig, eine Hash Value einer kompromittierenden Datei zu erzeugen und diese damit eindeutig zu beschreiben. Daher existieren eine Menge IoC, die Hash Values enthalten. Auf der anderen Seite ist es für die Ersteller der kompromittierenden Datei nicht schwer, die Hash Value zu verändern. Schon ein einziges hinzugefügtes Zeichen oder eine Bitverschiebung in nicht verwendeten Ressourcen sorgt dafür, dass sich die Hash Value ändert.

Jeder Angriff auf ein System, der über eine Netzwerkverbindung geschieht, hinterlässt dabei eine IP-Adresse, mit der unter Umständen Rückschlüsse auf den Angreifer getroffen werden können. Allerdings werden häufig Methoden, wie z.B. anonyme Proxy Dienste verwendet, um die eigene IP-Adresse zu verschleiern. Daher reicht es meistens nicht aus, eine einzige IP-Adresse zu blockieren, um sich vor einer bestimmten Malware oder einem bestimmten Angreifer zu schützen.

Ähnlich wie bei den IP-Adressen, können gelegentlich auch Domain Names bei der Übertragung von kompromittierenden Dateien auf das eigene System beobachtet werden. Domain Names müssen registriert werden und kosten in der Verwendung Geld. Daher ist es für die Angreifer mit Aufwand und auch Kosten verbunden, den Domain Name der Seiten zu ändern, über die kompromittierende Dateien verteilt werden.

Netzwerk Artefakte können z.B. URI Pattern, HTTP User Agenten oder SMTP Mailer Werte sein. Host Artefakte auf der anderen Seite können unter anderem Registry Keys/Werte, Ordner oder Dateien, die an bestimmten Orten erstellt wurden oder verdächtige Dienste sein, die nach einer Kompromittierung auf einem Computersystem aufgetaucht sind. Werden diese Artefakte durch einen IoC beschrieben, zwingt es den Angreifer, Zeit zu investieren und seine Angriffsmuster zu überarbeiten.

Teilweise benötigt von Angreifern verteilte Malware zusätzliche Programme, um ausgeführt werden zu können. Können diese Programme identifiziert und durch IoC beschrieben werden, kann, durch Blockierung der entsprechenden Programme, die Ausführung

der Malware unterbunden werden. Angreifer müssen sich in diesem Fall nach neuen Programmen umschauen, mit deren Hilfe sie ihre Malware zukünftig ausführen können.

TTPs (Tactics Techniques and Procedures) sind weniger Artefakte, die auf einem System gefunden werden können, sondern beschreiben eher die eigentlichen Methoden, die Angreifer verwenden, um ihre Angriffe durchzuführen. Spearfishing, mit in PDF Dateien enthaltenen Trojanern, ist ein Beispiel dafür. Gelingt es, diese TTPs erfolgreich durch z.B. Filterungsregeln zu unterbinden, ist der Angreifer gezwungen, nicht nur einzelne Artefakte oder Dateien seines Angriffs anzupassen, sondern die grundlegende Methode zu überdenken.

2.3 Correlation Process

Der Correlation Process wurde von Mitgliedern des Security Departmet of Computer Science von der Universität von Kalifornien entwickelt, um eine Übersicht über sicherheitsrelevante Aktivitäten innerhalb eines Netzwerks zu bekommen. Der Prozess enthält eine Sammlung von Komponenten, die Alarmierungen von Intrusion Detection Sensoren, in Intrusion Reports transformieren. Die Beschreibung der im restlichen Kapitel beschriebenen Komponenten, des in Abbildung 2.6 dargestellten Prozesses, orientiert sich an dem Bericht, den die Mitglieder des Security Department of Computer Science 2014 herausgegeben haben. [20]

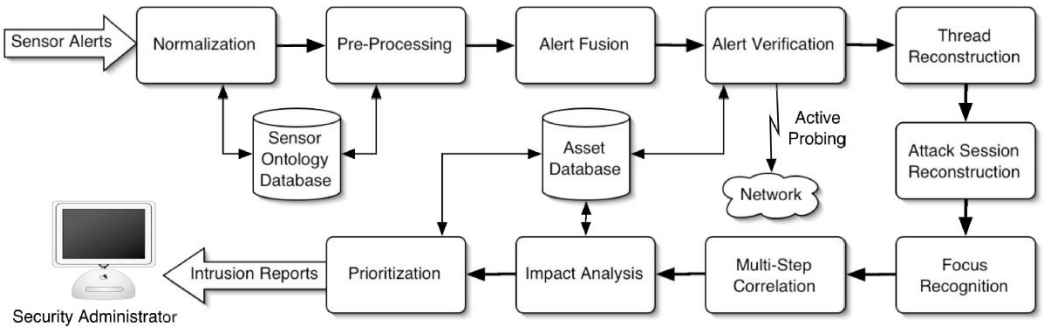

Abbildung 2.6: Correlation Process [20]

Im ersten Schritt übersetzt die Normalization (Normalisierung) Komponente eingehende Alarme in ein standardisiertes Format, welches von allen Komponenten verstanden wird.

Nachfolgend werden die normalisierten Alarme, in der Pre-Processing (Vorverarbeitung) Komponente, um weitere Attribute, wie z.B. die Start- und Endzeit, erweitert. Somit erhalten alle eingehenden Alarme, alle erforderlichen Attribute.

Die Alert Fusion Komponente(Alarm Fusionierung) kombiniert unabhängige Alarme von unterschiedlichen Sensoren, die zur selben Angriffsinstanz gehören.

Fehlgeschlagene Angriffe werden von der Alert Verification (Verifikation) Komponente markiert. Die Alarme, die fehlgeschlagene Angriffe enthalten, werden im restlichen Prozess mit geringerer Priorität behandelt.

Mehrere Alarme, die Angriffe eines Angreifers, auf ein einzelnes Ziel enthalten, werden von der Threat Reconstruction (Bedrohungsrekonstruktion) Komponente zusammengefasst.

Die Attack Session Reconstruction (Angriffsrekonstruktion) Komponente fasst host- und netzwerkbasierte Alarme zusammen, die sich auf denselben Angriff beziehen.

Die nächsten beiden Komponenten befassen sich mit Alarmen, die eine Vielzahl verschiedener Hosts betreffen. Die Focus Recognition (Fokus Erkennung) Komponente identifiziert die jeweiligen Hosts. Dabei werden sowohl die Hosts identifiziert, die die Quelle des Angriffes sind, als auch die Hosts, die das Ziel des Angriffes sind. Damit werden unter anderem Denial of Service (DOS) Angriffe identifiziert.
Die Multi Step Correlation (Abhängigkeit mehrstufiger Angriffe) Komponente hat die Aufgabe, bekannte Angriffsmuster zu erkennen. Die Angriffsmuster bestehen dabei aus einzelnen, zusammengesetzten Angriffen, die, über das Netzwerk verteilt, auftreten können.

Die Impact Analyse (Auswirkungsanalyse) Komponente ermittelt die Auswirkungen, die die entdeckten Angriffe auf den Betrieb des Netzwerkes und auf die betroffenen Systeme haben.

Anhand der Impact Analyse, vergibt die Priorization (Priorisierung) Komponente jedem Alarm eine eigene Priorität, mit dieser gefiltert werden kann, welcher Alarm Handlungsmaßnahmen erfordert und welcher möglicherweise ignoriert werden kann.

3 Konzipierung

Um das Ziel der Arbeit zu erreichen, wird ein System benötigt, auf dem sowohl die Honeypots, als auch der Prozess zur Strukturierung, Analyse und Formalisierung der aufgezeichneten Daten implementiert werden können.

Um möglichst viele Angriffe zu erhalten, wird das System im Internet platziert, wo es exponiert, ohne zusätzliche Sicherheitskomponenten, frei angreifbar ist. Um auf der anderen Seite die Risiken einer Kompromittierung und Übernahme des Systems zu verringern, werden lediglich Low- und Medium-Interaction Honeypots eingesetzt, deren Konfigurationsaufwand relativ gering ist. Außerdem bieten die geringen Interaktionsmöglichkeiten mit den Honeypots kaum, bis gar keine Möglichkeit, Zugriff auf das restliche System zu erlangen.

Damit die Funktion des Prozesses, zur Formalisierung und Standardisierung der durch die Angriffe generierten Daten, ausreichend getestet werden kann, werden zwei unterschiedliche Honeypotsensoren implementiert, deren Log-Daten sich voneinander unterscheiden.

3.1 Architektur

Es wird ein einzelnes Serversystem implementiert, auf dem sowohl die beiden Honeypotsensoren, als auch der Alarmierungsprozess implementiert werden. Das System soll über eine öffentliche IP-Adresse und einem Domänennamen verfügen, der für die Implementierung eines Web-Honeypots verwendet werden kann.

3.2 Alarmierungsprozess

Anlehnend an den in Kapitel 2.3 beschriebenen Correlation Process, wird zur Umsetzung des Ziels der Bachelorarbeit ein ähnlicher Alarmierungsprozess benötigt. Im Unterschied zum Correlation Process, sollen die Alarme seitens Honeypotsensoren erfolgen und nicht durch Intrusion Detection Systems. Des Weiteren liefern die Honeypotsensoren ihre Alarme an eine Instanz, die auf demselben System implementiert ist, wie die Sensoren auch. Deshalb können die Komponenten des Correlation Process, die für den Einsatz in einem Netzwerk entwickelt wurden, nicht eins zu eins übernommen werden.

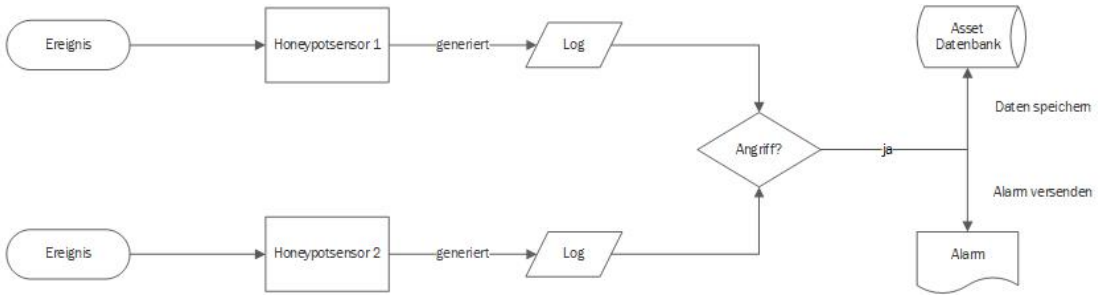

Abbildung 3.1: Ablaufprozess

Wird der reine Ablauf betrachtet, in dem die Ereignisse am System eintreffen und letztendlich eine Warnmeldung generiert wird, so entsteht der in Abbildung 3.1 dargestellte Prozess.

An beiden Honeypotsensoren geschehen Ereignisse, wodurch die Sensoren Log Daten generieren. Anhand dieser Log Daten muss bestimmt werden, ob ein Angriff vorliegt oder nicht. Zunächst müssen die Daten vergleichbar sein. Also wird ein Prozess benötigt, der die Daten in ein vergleichbares Format bringt. Außerdem muss genau festgelegt werden, wodurch sich ein Angriff definiert.

Wird ein solch definierter Angriff in den formalisierten Log-Daten erkannt, wird aus den Log-Daten ein Alarm generiert und versandt. Da zusätzlich zum Versenden des Alarms auch ein Indicator of Compromise erzeugt werden soll, der den entsprechenden Angriff beschreibt, sollte der Prozess zur Formalisierung der Daten auf einen, in Kapitel 2.2 erwähnten, IoC Standard abbilden.

Zusätzlich werden die Daten des Angriffs in einer Datenbank gespeichert, um bei zukünftigen Angriffen die Daten auf mehrfaches Auftreten untersuchen zu können.

Werden die Ergebnisse der Betrachtung des Ablaufprozesses auf den Correlattion Prozess übertragen, so treten die ersten drei Komponenten des Correlation Processes ebenso im zuvor analysierten Ablauf auf.

Die Normalisierungskomponente wird zusammen mit der Pre-Processing Komponente dafür benötigt, die unterschiedlichen Log-Daten der Honeypotsensoren, auf ein gemeinsames Format zu bringen. Außerdem benötigt dieses Format ein Mindestsatz an Attributen, um einem noch auszuwählendem IoC Standard zu entsprechen.

Abseits des in Abbildung 3.1 dargestellten Ablaufs, können auch weitere Komponenten des Correlation Processes in das Konzept des Alarmierungsprozesses mit aufgenommen werden.

Auch Angriffe aus derselben Instanz, die auf beiden Honeypotsensoren auftreten, sollen entsprechend identifiziert werden können, weshalb die Alert Fusion Komponente ebenfalls benötigt wird.

Nach der Identifizierung eines Angriffes, wird jeder Angriff gleichbehandelt und löst eine Warnmeldung aus. Dies gilt sowohl für fehlgeschlagene, als auch für erfolgreiche Angriffe. Daher müssen diese nicht unterschieden werden, was dazu führt, dass die Alert Verification Komponente nicht benötigt wird.

Es soll ebenfalls möglich sein, mehrere Angriffe eines Angreifers, auf einen einzelnen Sensor, im Nachhinein erkennen zu können und diese zusammenzufassen. Daher muss die Threat Reconstruction Komponente ebenfalls beachtet werden.

Die Attack Session Komponente, zur Zusammenfassung von host- und netzwerkbasierten Angriffen, wird auf Grund eines einzelnen eingesetzten Systems nicht benötigt.

Die Focus Recognition Komponente kann in der geplanten Umgebung dazu eingesetzt werden, um eine Vielzahl derselben Angriffe, die in kurzer Zeit auf demselben Sensor auftreten, zu identifizieren.

Um bekannte Angriffsmuster aus einer Vielzahl von einzelnen Angriffen zusammenzusetzen und zu identifizieren, bedarf es eines hohen Implementierungsaufwandes und einer hohen Komplexität, weshalb die Multi Step Correlation Komponente im Rahmen der Bachelorarbeit nicht beachtet wird.

Da es, abseits des eigentlichen Betriebssystems des eingesetzten Servers, keine schützenswerten Komponenten und kein schützenswertes Netzwerk gibt, werden die Impact Analyse und die Priorization Komponente nicht dafür benötigt, die auftretenden Ereignisse zu bewerten und zu priorisieren. Wie zuvor beschrieben, werden identifizierte Angriffe gleich behandelt.

Abbildung 3.2: Alarmierungsprozess

So ergibt sich aus den benötigten Komponenten, der in Abbildung 3.2 dargestellte Alarmierungsprozess, welcher im weiteren Verlauf der Bachelorarbeit mit Leben gefüllt werden muss.

3.3 Eingesetzte Honeypots

Als Honeypotsensor wird die Web-Honeypot Kombination aus den frei verfügbaren Projekten SNARE und TANNER verwendet. Zusätzlich wird der, ebenfalls frei verfügbare, SSH Sensor Cowrie eingesetzt.

3.3.1 SNARE/TANNER

Super Next generation Advanced Reactive honEypot (SNARE) ist ein Web-Honeypotsensor, der vom Honeynet Projekt entwickelt wurde und stetig weiterentwickelt wird. Bei Installation liefert SNARE eine vorgefertigte Website, auf der Angriffe eingehen sollen. Zusätzlich beinhaltet SNARE ein Tool, um bestehende Internetseiten zu klonen und diese anstelle der vorkonfigurierten Website zu verwenden.

TANNER wurde, ebenfalls vom Honeynet Projekt, entwickelt, um HTTP Anfragen, die auf dem SNARE Sensor eingehen zu analysieren und auf diese zu reagieren. Um die eingehenden Anfragen beantworten zu können und den SNARE Sensor mit Leben zu füllen, liefert TANNER eine Liste von so genannten Dorks.

Die Liste der Dorks basiert auf der Google Hacking Database und anderen Quellen, welche diverse Suchanfragen oder URL Pfade zu beliebten Verzeichnissen enthält. Werden also

Anfragen an den SNARE Honeypot gestellt, die die in der Liste der Dorks verzeichneten Anfragen enthält, reagiert TANNER auf diese Anfragen. [10, vgl.]

Zusätzlich kann TANNER eine Auswahl von Angriffsmustern und Schwachstellen, auf der von SNARE bereitgestellten Internetseite, emulieren. Dazu gehören unter anderem Remote File Inclusion, Cross Site Scripting und SQL Injection. [11, vgl.]

Die SNARE/TANNER Kombination bildet einen Low-Interaction Honeypotsensor. Die Interaktion mit dem Honeypot ist auf die HTTP Anfragen begrenzt, die über die URL eingehen. Die Komplexität und der Umfang der bereitgestellten Website trägt ebenfalls zur Begrenzung der Interaktion bei.

Um SNARE und TANNER auf dem Serversystem installieren zu können, werden folgende Komponenten benötigt: [7] [12]

- Linux Distribution
- Python3
- Redis
- Docker
- PHPox
- SNARE Internetseite

Eine eigens erstellte Internetseite ist zwar nicht zwingend erforderlich, da SNARE von Haus aus eine vorkonfigurierte Seite liefert, allerdings ist diese für jede SNARE Installation gleich und somit leicht als Honeypot zu identifizieren. Die zu erstellende Internetseite muss nicht über viele Funktionen verfügen. Es reicht aus, ein Login Formular zu haben, mit dem SQL Injection emuliert werden kann. Weitere Emulationen können über den URL Pfad erfolgen.

Eingehende Anfragen und Events werden von TANNER in einer lokal gespeicherten Log Datei abgelegt. Die Einträge in der Log Datei liegen im json Format vor. In den aufgezeichneten Daten sind Informationen über die eingesetzte Request Methode, den angeforderten Pfad, den HTTP-Header, die Quell IP-Adresse und Port und die versendete Antwortnachricht enthalten.

3.3.2 Cowrie

Cowrie ist ein Medium-Interaction SSH und Telnet Honeypot, welcher entwickelt wurde, um Brute Force Angriffe und die Interaktion eines Angreifers mit der Kommandozeile aufzuzeichnen. Durch eine Konfigurationsdatei kann festgelegt werden, welchen Kombinationen von Benutzernamen und Passwort der Zugang zum virtuellen Dateisystem gewährt wird. [2]

Nach erfolgreicher Anmeldung am Cowrie Sensor, können Angreifer auf einer virtuellen Umgebung mit einer Kommandozeile interagieren, die auf einem emulierten Dateisystem läuft. Das Dateisystem, sowie eine Anzahl von Befehlen für die Kommandozeile, sind in Cowrie vorgefertigt.

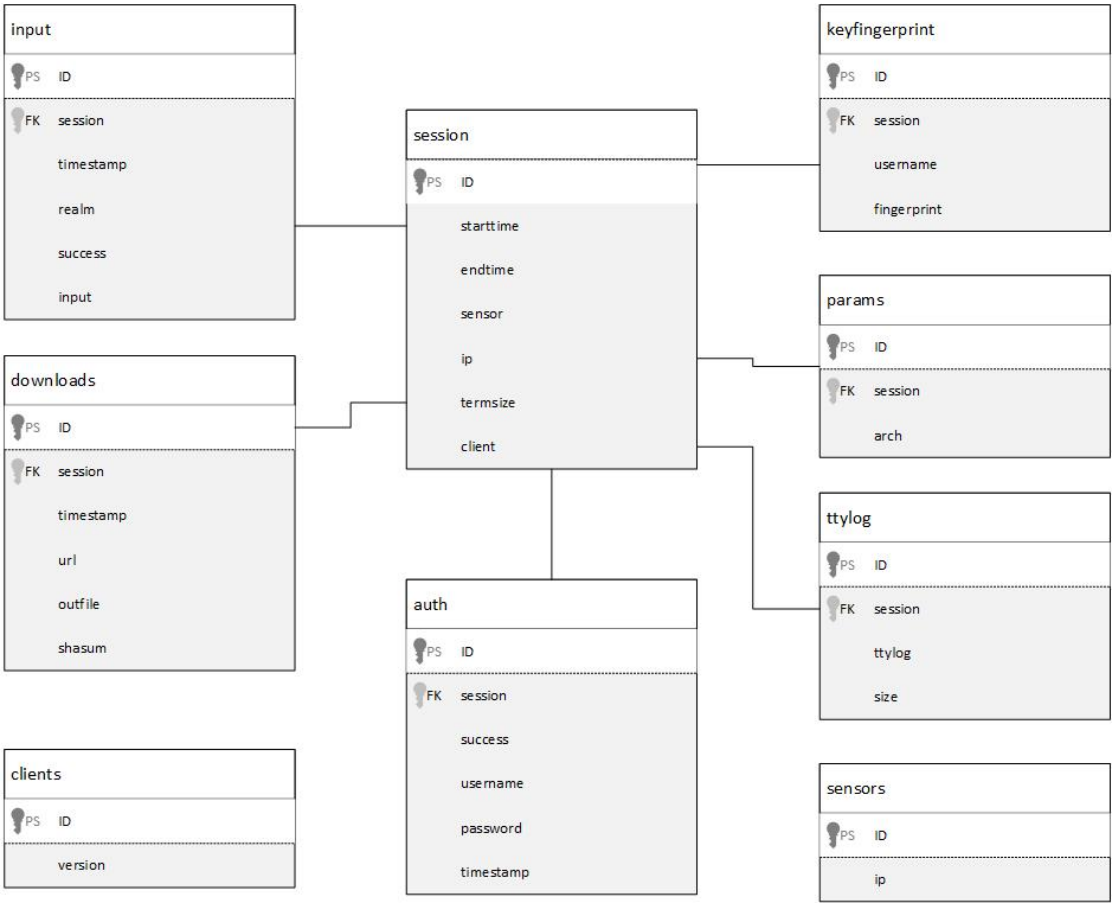

Abbildung 3.3: Cowrie Log Datenbank

Sollten, während der Kommandozeileninteraktion einer Sitzung, Dateien heruntergeladen werden, speichert Cowrie diese in einem separaten Verzeichnis. Somit können diese Dateien im Nachhinein analysiert werden.

Jeder Login Versuch am Sensor, sowie jede Interaktion mit der Kommandozeile, die nach einem erfolgreichen Login erfolgt, wird aufgezeichnet. Cowrie speichert die aufgezeichneten Daten in einer MySQL Datenbank. Das Schema der Datenbank ist in Abbildung 3.3 dargestellt.

Für die Installation von Cowrie werden folgende Komponenten benötigt: [2]

- Linux Distribution
- Python3.5
- python-virtualenv

3.4 Eingesetzte Standards

Damit im Zuge der Bachelorarbeit erzeugte Alarme allgemein verständlich sind und später gegebenenfalls weiterverarbeitet und genutzt werden können, sollen diese Alarme in einem standardisierten Format erfolgen. Dies betrifft sowohl die vom Honeypot generierten Alarme, als auch die in den Alarmen beschriebenen Angriffsmuster.

Da erzeugte Alarme in Form von Indicators of Compromise erfolgen sollen, wird der bereits in Kapitel 2.2 kurz erwähnte Standard STIX eingesetzt.

Auch die Angriffe und Angriffsmuster sollen allgemeinverständlich sein. Daher wird bei der Umsetzung der Bachelorarbeite zum Beschreiben der einzelnen Angriffe und Angriffsmuster der CAPEC Standard verwendet. Auch dieser wird im weiteren Verlauf dieses Kapitels genauer beleuchtet.

3.4.1 STIX (Structured Threat Information eXpression)

Structured Threat Information eXpression (STIX) ist ein frei verfügbares Format, zur Serialisierung und zum Austausch von CTI. STIX wurde von der Non Profit Organisation OASIS entwickelt, welche die Entwicklung von offenen Standards, in der globalen Informationsgesellschaft, vorantreibt. [8, vgl.]

STIX setzt sich aus 12 Objektentitäten zusammen, mit denen Daten dargestellt und miteinander in Verbindung gesetzt werden können.

Die für diese Bachelorarbeit relevante Objektentitäten sind Observed Data und Attack Pattern.

Observed Data bietet die Möglichkeit über die Honeypotsensoren gesammelte Daten, in STIX Data Objects genannt, wie unter anderem IP-Adressen, Hashes oder HTTP-Requests zu überwachen und mit weiteren Attributen zu verknüpfen. Ein Datensatz eines Observed Data Objektes enthält eine Liste, der jeweils überwachten Data Object, mit dem ersten und letzten Zeitpunkt ihrer Sichtung und der Anzahl der Sichtungen.

Mit der Objektentität Attack Pattern lassen sich Angriffe klar definieren, um sie später eindeutig zuordnen zu können. Zusätzlich besteht die Möglichkeit, die Angriffsmuster über externe Referenzen beschreiben zu lassen.

Zusammen mit den restlichen Objektentitäten wurde STIX dazu entwickelt, Daten in größerem Umfang miteinander in Beziehung zu setzen. So können mit dem Objektmodell von STIX beispielsweise ganze Angriffskampagnen dargestellt werden, Informationen über einzelne Angreifer, sowie Informationen über Gruppen von Angreifern gesammelt werden. Aus der STIX Dokumentation ist jeweils zu entnehmen, welche Objektentität in welcher Form mit anderen Entitäten in Beziehung steht. In Abbildung 3.4 ist dies am Beispiel der Objektentität Threat Actor zu sehen.

Über eine von STIX bereitgestellte Python Bibliothek, können relevante Daten in STIX Objekte und somit in das STIX Format gebracht, in Beziehung gesetzt und ausgetauscht werden. In das STIX Format gebrachte Daten, werden im json Format dargestellt und auf dem Dateisystem des Systems gespeichert.

Im Rahmen der Bachelorarbeit wird STIX eingesetzt, da es über eine herausragende Dokumentation verfügt und den Vorteil bietet, dass es trotz seiner Komplexität modular einsetzbar ist und somit auf die individuellen Anforderungen angepasst werden kann.

Abbildung 3.4: Threat Actor [9]

Außerdem ist STIX komplett kompatibel zu dem Format TAXII, welches ebenfalls von OASIS entwickelt wurde und für den Austausch von CTI zuständig ist. Ein weiterer Vorteil ist die Python Bibliothek, über die das Format genutzt werden kann.

Begrenzt wird STIX durch die statischen Objekte, die so genutzt werden müssen, wie sie durch das Format vorgegeben werden. Es ist also nicht möglich, bei Bedarf weitere Attribute hinzuzufügen, die unter Umständen benötigt werden, um analysierte Daten wie gewünscht darzustellen.

3.4.2 CAPEC (Common Attack Pattern Enumeration and Classification)

Common Attack Pattern Enumeration and Classification (CAPEC) stellt einen öffentlich zugänglichen Katalog von Angriffsmustern bereit. Angriffsmuster sind Beschreibungen von üblichen Vorgehensweisen, die Angreifer dazu benutzen, um auf Basis bekannter Schwachstellen Systeme zu kompromittieren. [1, vgl.]

Jeder Eintrag im CAPEC Katalog liefert eine Vielzahl verschiedener Informationen über das jeweilige Angriffsmuster. Es sind unter anderem Informationen über den Ablaufprozess, die benötigten Fähigkeiten zur Ausführung und aktuelle Schwachstellen des jeweiligen Angriffsmusters enthalten.

Zur Identifizierung wird jedem Eintrag eine eigene ID zugewiesen. Zusätzlich enthalten CAPEC Einträge Verweise auf ähnliche oder zusammengehörige Angriffsmuster.

Die STIX Objektentität Attack Pattern unterstützt CAPEC IDs als externe Referenz, weswegen CAPEC im Rahmen der Bachelorarbeit dazu genutzt werden kann, Ereignisse, die auf den Honeypotsensoren passieren, als Angriffe zu identifizieren und diese formal mit Hilfe von STIX zu beschreiben.

Um den Alarmierungsprozess testen zu können, wird pro Honeypotsensor ein Angriffsmuster implementiert. Anhand der aufgezeichneten Log-Daten bietet sich für den Cowrie Sensor die Implementierung des Angriffsmusters Password Brute Forcing (CAPEC-49) an. Dieses Angriffsmuster zeichnet sich dadurch aus, dass innerhalb einer einzigen Sitzung, auf dem angegriffenen Sensor, eine Vielzahl von Loginversuchen ausgeführt werden. Dabei werden eine Vielzahl von Passwörtern, zu einem einzigen Benutzernamen, ausprobiert. [5]

Für den Web-Honeypot wird das Angriffsmuster Remote File Inclusion (CAPEC-253) implementiert. Die SNARE/TANNER Kombination bietet einen entsprechenden Emulator für dieses Angriffsmuster an. Bei Ausführung dieses Angriffsmusters, wird ein Dienst über eine enthaltene Schwachstelle dazu gebracht, ausführbare Dateien von einer externen Quelle herunterzuladen. Der Angreifer wird nach dem Herunterladen von entsprechenden Dateien versuchen, diese auf dem angegriffenen System auszuführen. [6]

3.5 Ergebnis der Konzipierung

Um die durch die Honeypotsensoren aufgezeichneten Daten, gemäß des in Kapitel 3.2 erstellten Alarmierungsprozesses verarbeiten zu können, müssen diese, nachdem sie in das STIX Format normalisiert wurden, abgespeichert werden, um zukünftig weiter verwendet werden zu können.

Dazu muss eine automatisierte Routine geschaffen werden, die die jeweiligen Log-Dateien der Honeypotsensoren regelmäßig auf passende CAPEC Angriffsmuster überprüft und die Daten, bei einem identifizierten Angriff, für eine weitere Verwendung normalisiert in eine Datenbank schreibt.

Anschließend muss, aus den normalisierten Daten des Angriffes, eine Warnmeldung generiert werden, welche per Mail versandt wird.

Die Datenbank, im folgenden Asset Datenbank genannt, die zur Speicherung der analysierten Daten benötigt wird, orientiert sich an den beiden STIX Objektentitäten Observed Data und Attack Pattern und enthält eine Tabelle, für jede der beiden Entitäten. Zusätzlich enthält das in Abbildung 3.5 dargestellte Datenbankschema die Verbindungstabelle Object Link. Mit Hilfe dieser Tabelle können weitere Tabellen für Data Objects, wie IP-Adressen und URLs, bei Bedarf erstellt und an die Observed Data Tabelle angebunden werden.

Die einzelnen Schritte, des vorher beschriebenen Ablaufs, werden jeweils über die in Abbildung 3.6 dargestellten Python Module realisiert.

So genannte CAPEC Module überprüfen die Log Daten der Honeypotsensoren auf aufgetretene Angriffsmuster. Dabei wird pro, in CAPEC gelistetem Angriffsmuster, ein eigenes Python Modul benötigt, da die Daten je nach Angriffsmuster individuell analysiert werden müssen. Entdeckt ein solches CAPEC Modul einen auf ihn zutreffenden Angriff, werden Module aufgerufen, die die Log-Daten des Angriffs in der Asset Datenbank speichern.

Abbildung 3.5: Asset Datenbank Schema

Zusätzlich rufen die Datenbankmodule ein Modul auf, welches aus den abgespeicherten Daten, mit Hilfe der STIX Python Bibliothek, Observed Data Objekte erzeugt. Die Objekte werden auf dem Dateisystem der Linux Distribution gespeichert. Sie fungieren als Indicator of Compromise und können per Mail als Warnmeldung versandt werden.

Die CAPEC Module werden in regelmäßigen Intervallen durch einen Cronjob ausgeführt.

Werden nun die Asset Datenbank und die Nutzung der einzelnen Python Module mit dem Alarmierungsprozess zusammengeführt, ergibt sich daraus der in Abbildung 3.7 dargestellte Prozess.

Wird durch die, durch den Cronjob ausgeführten, CAPEC Module ein Angriffsmuster erkannt, wird die Asset Datenbank überprüft. Sollten die Daten, die im Zuge des Angriffes aufgezeichnet wurden, bereits in der Asset Datenbank, durch einen Observed Data Datensatz überwacht werden, werden die entsprechenden Datensätze und die dazugehörigen, auf dem Dateisystem gespeicherten, Observed Data Objekte aktualisiert. Sollten noch keine Datensätze in der Datenbank vorliegen, werden diese, zusammen mit den zugehörigen Observed Data Objekten, neu erzeugt.

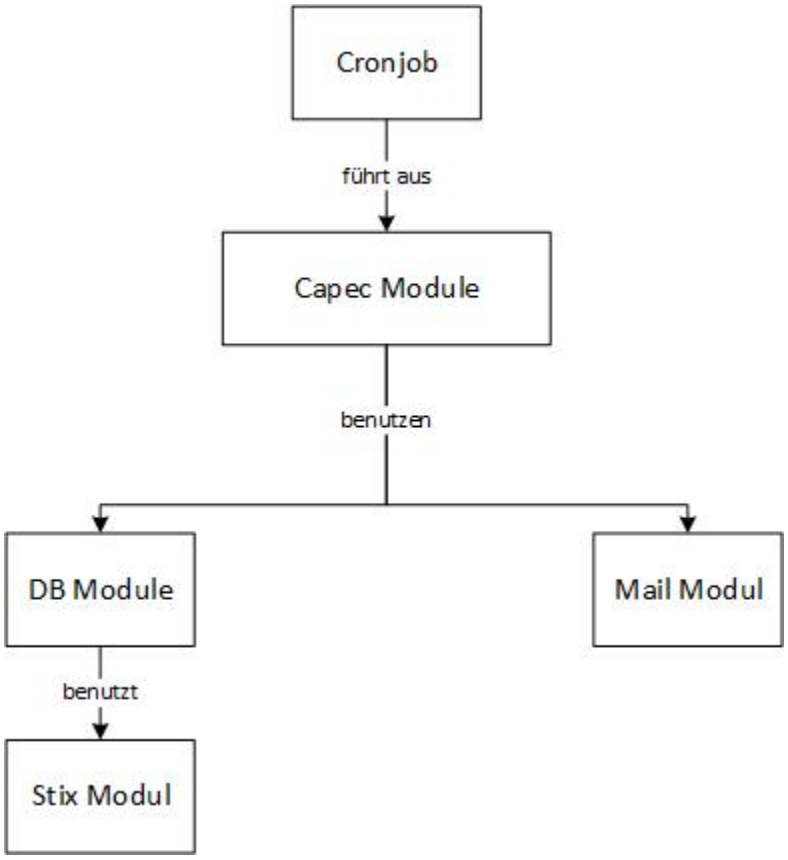

Abbildung 3.6: Benutzung der Python Module

Anschließend werden die Observed Data Objekte mit dem zugehörigen Attack Pattern in einem STIX Bundle verknüpft und per Mail als Warnmeldung verschickt. Ein STIX Bundle fasst eine Anzahl von STIX Objekten in einer Sammlung zusammen und stellt diese zusammen dar.

Die relevanten Komponenten des Correlation Processes, die in Kapitel 3.2 identifiziert wurden, werden über die eingesetzten Python Module und die Komponenten des in Abbildung 3.7 dargestellten Prozesses umgesetzt. Die Normalization und das Pre-Processing werden mit Hilfe der Datenbankmodule umgesetzt. Durch die Speicherung der analysierten Daten in der Asset Datenbank, sind die Daten um alle benötigten Parameter erweitert und im selben Format dargestellt.

Die Alert Fusion Komponente, die Angriffe der selben Instanz, die auf unterschiedlichen Sensoren geschehen, verarbeitet, wird bei Überprüfung bereits in der Asset Datenbank gespeicherter Daten umgesetzt.

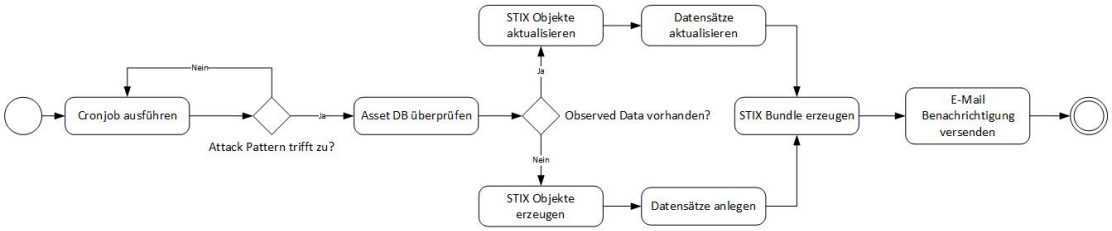

Abbildung 3.7: Alarmierungsprozess Version 2

Zwar bietet der STIX Standard die Möglichkeit, Informationen über Angreifer zu sammeln und darzustellen, allerdings können die dazu erforderlichen Daten nicht über die eingesetzten Honeypotsensoren gesammelt werden. Um die STIX Objektentität Threat Actor nutzen zu können, wird wie in Abbildung 3.4 dargestellt, der Einsatz weiterer Objektentitäten benötigt. Auf Grund des Umfangs der Umsetzung findet daher die Identifizierung der Angreifer und somit die Threat Reconstruction Komponente im Rahmen der Bachelorarbeit keine Beachtung. In Weiterführenden Arbeiten können die erforderlichen Objektentitäten allerdings an das bestehende Datenbankschema angebunden werden.

Wie in Kapitel 3.2 festgestellt, würde die Focus Recognition Komponente bei dem Einsatz eines einzigen Servers nur zum Einsatz kommen, wenn eine Vielzahl von angreifenden Hosts an einem einzigen Angriff, wie z.B. einem Denial of Service Angriff, beteiligt sind. Dabei würde diese Komponente durch das zugehörige CAPEC Modul bearbeitet werden. Allerdings wird ein CAPEC Modul, zur Erkennung eines entsprechenden Angriffsmusters, im Rahmen der Bachelorarbeit nicht implementiert.

4 Umsetzung und Test

Bevor es an die Programmierung, der in Kapitel 3.5 vorgestellten Python Module, zur Umsetzung des Alarmierungsprozesses geht, müssen die Serverumgebung, sowie die Honeypotsensoren installiert und konfiguriert werden. Auch die Assetdatenbank muss vorher implementiert werden, damit die Python Module darauf zugreifen können.

Im Anschluss an die Implementierung der benötigten Komponenten, wird in diesem Kapitel die Funktionalität der Automation des Alarmierungsprozesses, anhand der umgesetzten Angriffsmuster, exemplarisch getestet.

4.1 Installation der Serverumgebung

Die Serverumgebung wurde, zusammen mit einer öffentlichen IP-Adresse und einem Domänennamen für die Internetseite des SNARE Web-Honeypots, bei einem Anbieter für Web-Hosting und Serversysteme gemietet.

Das Serversystem kommt vorkonfiguriert mit der Linux Distribution Debian Jessie. Auf das System kann mit root Rechten, über SSH oder eine Konfigurationsseite im Internet zugegriffen werden. Somit kann das System nach Belieben konfiguriert werden.

Betriebssystem	Debian Jessie
IP-Adresse	37.120.187.182
Domänenname	www.rasen-ba.de

Tabelle 4.1: Serversystem

4.2 Installation und Konfiguration von SNARE/TANNER

Damit SNARE/TANNER auf dem Serversystem installiert werden können, werden zunächst die in Kapitel 3.3.1 aufgelisteten Komponenten installiert. Im Anschluss daran, wird die Web-Honeypot Kombination, nach der auf der jeweiligen Projektseite beschriebenen Installationsanleitung, installiert. [12] [7]

Der unten stehende Code Block zeigt den Code der Internetseite, die für den Betrieb des SNARE Sensors erforderlich ist. Die Seite verfügt über jeweils ein Textfeld, für die Eingabe eines Benutzernamen und eines Passworts.

```
<body background="back.jpg">
<form method = "post" action="welcome.php">
<h1 align="center" style = "color:white">Very vulnerable</h1>
<div class="container">
<label for="uname" style = "color:white"><b>Username</b></label>
<input type="text" placeholder="Enter Username"
name="uname" required>
<label for="psw" style = "color:white"><b>Password</b></label>
<input type="password" placeholder="Enter Password"
 name="psw" required>
<button type="submit">Login</button>
 </form>
</body>
```

Bevor diese Seite für den Einsatz des Honeypotsensors genutzt werden kann, muss sie über das von SNARE mitgelieferte Tool zum Klonen von Internetseiten geklont werden. Dazu wird sie vorübergehend über Port 80 des Servers veröffentlicht und über das Kommando *sudo clone –target http://rasen-ba.de* geklont.

Die geklonte Internetseite wird auf dem Dateisystem des Servers gespeichert und kann im Anschluss zum Betreiben des SNARE Sensors genutzt werden.

SNARE ist nach der Erstellung und dem Klonen der Internetseite einsatzbereit. Um die volle Funktionalität der Web-Honeypot Kombination nutzen zu können, muss vor Inbetriebnahme TANNER konfiguriert werden. Zur Konfiguration von TANNER dient eine vorgefertigte Konfigurationsdatei, die auf die eigenen Anforderungen angepasst werden kann.

Die Konfigurationsdatei ist in einem ini ähnlichem Format geschrieben. Für den Einsatz im Rahmen der Bachelorarbeit, werden nur wenige Konfigurationsparameter benötigt. So muss definiert werden, auf welchem Host und Port die Dienste PHPOX, REDIS und TANNER laufen sollen. In der Tabelle 4.2 sind die jeweiligen Einstellungen gelistet. Des Weiteren wird eine Angabe der aktivierten Emulatoren benötigt.

Dienst	Host	Port
TANNER	localhost	8090
PHPOX	localhost	8088
REDIS	localhost	7777

Tabelle 4.2: TANNER Konfiguration

Wie in Kapitel 3.4.2 beschrieben, wird der Einsatz von SNARE und Tanner auf CAPEC-253 beschränkt, weshalb lediglich der Emulator für Remote File Inclusion aktiviert werden muss.

```
1 [EMULATOR_ENABLED]
2 sqli=False
3 rfi=True
4 lfi=False
5 xss=False
6 cmd_exec=False
7 php_code_injection = False
8 php_object_injection= False
9 crlf=False
10 xxe_injection= False
```

Zum Schluss wird noch angegeben, unter welchem Verzeichnis die Log-Daten gespeichert werden sollen, die durch die Analyse, der auf den Honeypotsensoren eingehenden Ereignissen, aufgezeichnet werden.

```
1 [LOCALLOG]
2 enabled=True
3 PATH=/tmp/tanner_report.json
```

Nachdem alle Konfigurationen für SNARE und TANNER abgeschlossen sind, können die erforderlichen Dienste der Reihe nach gestartet werden. Dabei ist darauf zu achten, dass die Dienste nach ihrem Start, im Hintergrund weiter ausgeführt werden, und auch nach Beendung der Sitzung mit dem Server, weiter ausgeführt werden. Nachdem alle erforderlichen Dienste gestartet wurden, kann zunächst TANNER und im Anschluss daran SNARE den Betrieb aufnehmen.

SNARE stellt die zuvor erzeugte Internetseite über Port 80 des Servers zur Verfügung. TANNER zeichnet die Log-Daten der eingehenden Ereignisse auf. Die Log-Daten müssen im weiteren Verlauf dieser Arbeit noch auf das Angriffsmuster CAPEC-253 analysiert und entsprechend verarbeitet werden.

4.3 Installation und Konfiguration von Cowrie

Neben den in Kapitel 3.3.2 aufgelisteten Komponenten, die für eine Installation von Cowrie benötigt werden, müssen noch weitere Schritte vor Inbetriebnahme des Sensors ausgeführt werden. So wird auf der Dokumentationsseite von Cowrie ausdrücklich dazu geraten, einen eigenen Benutzer für die Ausführung von Cowrie anzulegen, der über keine root Berechtigungen verfügt. [3]

Da mit dem SSH Sensor möglichst viele Angriffe aufgezeichnet werden sollen, wird Cowrie über den standard SSH Port 22 betrieben. Daher wird der SSH Zugang zum Server auf den Port 2222 umgeleitet.

Im Rahmen der Bachelorarbeit wird Cowrie als reiner SSH Honeypot betrieben. Daher muss der Betrieb von Telnet in der Konfigurationsdatei von Cowrie deaktiviert werden. Zusätzlich müssen die folgenden Einstellungen in der Konfigurationsdatei vorgenommen werden, damit die Daten eingehender SSH Verbindungen in eine MySQL Datenbank gespeichert werden.

```
1 [output_mysql]
2 enabled = true
3 host = localhost
4 database = cowrie
5 username = cowrie
6 password = wUtgKvaHutN7t4m5Owsi
7 port = 3306
```

4.4 Erstellung der Asset Datenbank

Die Asset Datenbank wird, anhand des in Abbildung 3.5 dargestellten Datenbankschemas, über ein SQL Skript erzeugt. Zusätzlich zu den im Datenbankschema dargestellten Entitäten, werden weitere Entitäten angebunden, die für die Umsetzung der beiden zu implementierenden CAPEC Angriffsmuster benötigt werden. Dabei wird an die Object Link Tabelle die Entität Data Object angebunden. Diese fungiert als Elterntabelle für die einzelnen Entitäten der jeweiligen Daten.

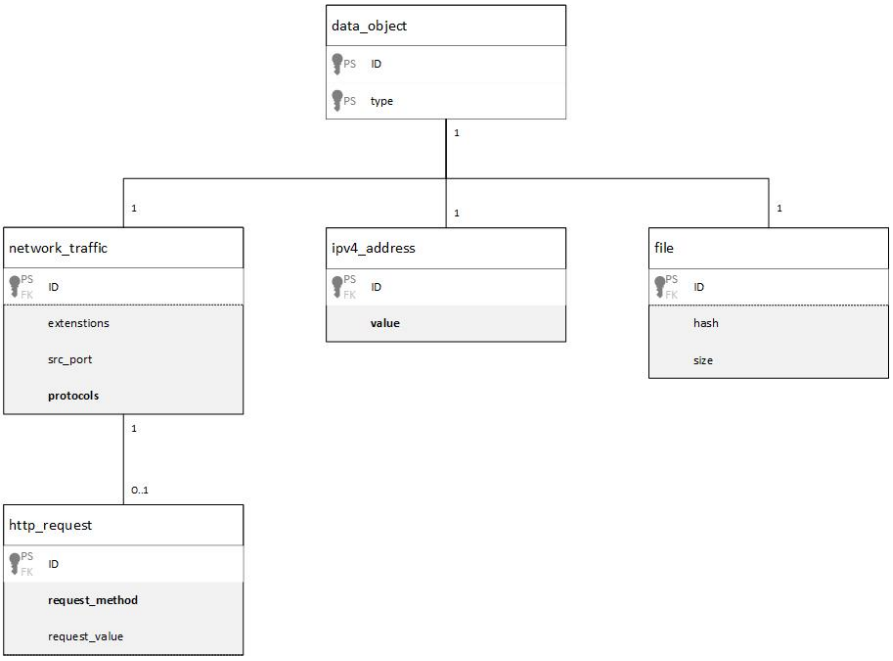

Abbildung 4.1: Data Object Entitäten

Die Abbildung 4.1 zeigt alle Entitäten, die für die Umsetzung der Angriffsmuster benötigt werden. Für das Password Brute Forcing Angriffsmuster wird die jeweilige IP-Adresse des Angreifers aufgezeichnet. Die analysierten Daten der Remote File Inclusion beinhalten ebenfalls die Quell IP-Adresse des Angreifers. Zusätzlich werden der Hash Wert und die Größe der eingeschleusten Datei aufgezeichnet. Außerdem werden die Daten der Netzwerkverbindung, sowie relevante Daten des HTTP Requests gespeichert.

4.5 Programmierung des STIX Moduls

Das STIX Modul *stix.py* wird von den Datenbankmodulen verwendet, um zu den in der Asset Datenbank aufgezeichneten Daten, STIX Objekte zu erzeugen. Die STIX Objekte werden im weiteren Prozess innerhalb einer Warnmeldung als IoC versandt.

Das Modul besteht wesentlich aus den drei Funktionen *add_observed*, *update_observed* und *create_bundle*. Die restlichen drei Funktionen *db_connect*, *db_disconnect* und *get_objects* dienen als Hilfsfunktionen.

Die Funktionen *db_connect* und *db_disconnect* stellen eine Verbindung mit der Asset Datenbank her-, beziehungsweise trennen die Verbindung. Diese Funktionen sind in allen implementierten Modulen vorhanden.

Die Funktion *add_observed* erstellt ein STIX Observed Data Objekt, speichert dieses als *json* Datei auf dem Dateisystem des Servers und gibt das Objekt an die aufrufende Funktion zurück. Als Übergabeparameter wird eine Liste von IDs benötigt, über die zu überwachende Data Objects identifiziert werden können. Zusätzlich wird der Zeitpunkt der Sichtung der zu überwachenden Daten übergeben.

```
1  #Ausschnitt aus add aus stix.py
2  #STIX Objekt erzeugen
3  ob1 = ObservedData(first_observed = time,
4      last_observed = time,
5      number_observed = 1,
6      objects = data_objects)
7
8  #STIX Objekt speichern
9  fs = FileSystemStore("../stix/objekte")
10 fs.add(ob1)
```

Über die importierte STIX Python Bibliothek können die Observed Data Objekte, wie in dem Code Ausschnitt dargestellt, erzeugt und auf dem Dateisystem gespeichert werden. Die Data Objects, die über *objects = data_objects* an das Observed Data Objekt übergeben werden, sind in einer Liste enthalten, die über die Hilfsfunktion *get_objects* geliefert wird.

Die Hilfsfunktion *get_objects* benötigt bei Aufruf eine Liste von Data Object IDs, deren Daten sie zurückgeben soll. Die Daten werden in Form eines Dictionarys an die aufrufende Funktion zurückgegeben. Da sich die Datensätze, der von der Tabelle Data Object erbenden Entitäten, sich unterscheiden, müssen sie, bei Rückgabe an die aufrufende Funktion, unterschiedlich behandelt werden.

Zunächst wird, anhand der übergebenen Liste, der jeweilige Typ der Datensätze aus der Asset Datenbank ermittelt und in einer Liste gespeichert. Anschließend wird über eine for Schleife jeder Datensatz der erzeugten Liste individuell in das Rückgabe Dictionary geschrieben.

```
1 #Ausschnitt aus get_objects aus stix.py
2 for type in type_list:
3         if(type == 'ipv4-addr'):
4     ...
5         elif(type == 'http-request'):
6     ...
7         elif(type == 'file'):
8     ...
```

Bestehen bereits Datensätze und Observed Data Objekte für eingehende Angriffe, werden diese über die Funktion *update_observed* aktualisiert. Dazu wird das zu aktualisierende Observed Data Objekt aus dem Dateisystem geladen. Anschließend werden die Anzahl der Sichtungen, sowie der Zeitpunkt der letzten Sichtung und der Modifizierung des Observed Data Objektes aktualisiert. Zur Aktualisierung des Objektes werden als Übergabeparameter der Zeitpunkt des entsprechenden Angriffes, sowie die STIX ID des zu aktualisierenden Objektes benötigt.

Über *create_bundle*, die letzte Funktion des STIX Moduls, werden STIX Objekte zu einem Satz zusammengefasst. So werden die erzeugten Observed Data Objekte mit den Angriffsmustern verknüpft, bei denen sie aufgezeichnet wurden. Dieser Satz von STIX Objekten wird als IoC per E-Mail versandt.

Als Übergabeparameter wird eine Liste von STIX IDs benötigt, über die die einzelnen Observed Data Objekte, die auf dem Dateisystem erzeugt wurden, eindeutig identifiziert werden können. Über den folgenden Code Ausschnitt werden die entsprechenden Objekte vom Dateisystem aufgerufen und in einer Liste gespeichert. Aus der erstellten Liste wird ein STIX Bundle erzeugt, welches an die aufrufende Funktion zurückgegeben wird.

```
#Ausschnitt aus create_bundle aus stix.py
#Stix Objekte in eine Liste laden
fs = FileSystemStore("../stix/objekte")
stix_objects = []

for id in stix_id:
      stix_objects.append(fs.get(id))

#Bundle erzeugen und speichern
bundle = Bundle(stix_objects)
```

4.6 Programmierung der Datenbankmodule

Es werden zwei Module implementiert, die die aufgezeichneten Daten der eingehenden Angriffe, in der Asset Datenbank speichern. Dazu wird je ein Modul für die Entität Observed Data und ein Modul für die Entität Data Object und alle von ihr erbenden Entitäten programmiert. Die Entität Attack Pattern wird über kein eigenes Modul verwaltet. Dort enthaltende Angriffsmuster werden manuell als Datensatz angelegt.

4.6.1 Data Object

Das Python Modul *data_object.py* verfügt für jede Kindtabelle von Data Object über eine Funktion, zum Hinzufügen und zum Auslesen eines Datensatzes, sowie eine Funktion, um die ID eines Datensatzes auszulesen. Zusätzlich verfügt das Modul über die beiden, im Rahmen des STIX Moduls bereits erläuterten Funktionen, zum Herstellen und Trennen einer Verbindung mit der Asset Datenbank. Außerdem ist die Funktion *get_fid* als Hilfsfunktion enthalten. Sie gibt die nächste freie ID der Data Object Tabelle an die aufrufende Funktion zurück.

Zum Anlegen eines Datensatzes wird die Funktion *add* aufgerufen. Als Übergabeparameter wird die nächste freie ID der Data Object Tabelle übergeben, die über die Funktion *get_fid* ausgelesen werden kann. Des Weiteren sind der Typ des anzulegenden Datensatzes, sowie eine Liste der zu speichernden Werte zu übergeben. Zunächst wird ein Datensatz in der Elterntabelle Data Object angelegt. Im Anschluss wird, über die jeweils individuelle Funktion, ein Datensatz in einer der Kindtabellen angelegt.

```
1  #Ausschnitt aus add aus data_object.py
2  #Typ auslesen und entsprechenden Typ-Datensatz anlegen
3  scursor.execute("insert into data_object values(%s, %s)",
4                  (id, type))
5
6  if(type == 'ipv4-addr'):
7        add_ip(id, value)
8  elif(type == 'http-request'):
9        add_http(id, value)
10 elif(type == 'file'):
11       add_file(id, value)
```

Die Funktionen zum Auslesen von Datensätzen aus den Kindtabellen, bekommen bei Aufruf die ID des auszulesenden Datensatzes übergeben. Über eine Abfrage in der Asset Datenbank, werden die Werte des Datensatzes in einer Liste gespeichert und zurückgegeben.

4.6.2 Observed Data

Das Python Modul *observed_data.py* überwacht Data Objects, die während eines Angriffes aufgezeichnet wurden. Das Modul übernimmt die Aufgabe des Pre-Processings, indem es die aufgezeichneten Daten mit weiteren Parametern, wie dem Zeitpunkt der ersten, beziehungsweise letzten Sichtung und der Anzahl der Sichtungen, verknüpft. Dazu verfügt das Modul über die Funktionen *add* , *update*, *get_id* und *get_stixid*.

Der Funktion *add* wird die ID der zu verknüpfenden Data Objects, sowie der Zeitpunkt des Angriffes übergeben. Über den Aufruf der Funktion *add_observed* des STIX Moduls wird zunächst ein Observed Data Objekt auf dem Dateisystem angelegt. Anschließend wird ein Datensatz in der Tabelle Observed Data erzeugt. Um den angelegten Datensatz

mit dem Data Object Datensatz zu verknüpfen, wird abschließend ein Eintrag in der Verbindungstabelle Object Link angelegt.

Werden durch Observed Data Objekte überwachte Datensätze erneut während eines Angriffes aufgezeichnet, werden die jeweiligen Objekte auf dem Dateisystem, sowie die dazugehörigen Einträge in der Asset Datenbank über die Funktion *update* aktualisiert. Zur Aktualisierung benötigt die Funktion die ID des zu aktualiserenden Observed Data Objektes, sowie den Zeitpunkt des Angriffes. Über die Hilfsfunktion *get_stixid* wird die STIX ID des Objektes ausgelesen. Über die STIX ID kann das Observed Data Objekt auf dem Dateisystem, mit Hilfe der Funktion *update_observed* des STIX Moduls, aktualisiert werden. Zusätzlich wird der zugehörige Datensatz der Asset Datenbank ebenfalls aktualisiert.

Die Funktion *get_id* bekommt bei Aufruf die ID eines Data Object Datensatzes übergeben, zu dem die Observed Data ID ausgelesen werden soll. Über eine SQL Abfrage wird die gesuchte Observed Data ID ausgelesen und an die aufrufende Funktion zurückgegeben.

4.7 Programmierung des Mail Moduls

Das Mail Modul wird von den CAPEC Modulen aufgerufen, um zuvor erzeugte STIX Bundle als IoC innerhalb einer Warnmeldung zu versenden. Das Modul verfügt über eine einzige Funktion *send_mail*. Innerhalb dieser Funktion wird eine Verbindung zum Mail Server des Systems aufgebaut, eine E-Mail Nachricht zusammengesetzt und versendet. Als Übergabeparameter wird der Funktion das zu versendende STIX Bundle übergeben.

```
1 msg = MIMEMultipart()
2 msg['From'] = "alert@rasen.rasen-ba.de"
3 msg['To'] = "lars92schneider@gmail.com"
4 msg['Subject'] = "Security Breach"
5
6 Smsg.attach(MIMEText(message.__str__()))
```

Die E-Mail Nachricht wird in dem dargetellten Code Ausschnitt zusammen gesetzt. Zum Ende des Auschnittes wird das übergebene STIX Bundle in das für die E-Mail Nachricht erforderliche Format geschrieben und an die Nachricht angehängt.

4.8 Programmierung der CAPEC Module

Anders als die zuvor programmierten Module, verfügen die beiden CAPEC Module über keine Funktionen. Sie arbeiten Schrittweise den in Kapitel 3.5 entwickelten, beziehungsweise in Abbildung 3.7 dargestellten, Alarmierungsprozess ab. Über einen Eintrag im Crontab des Servers, werden die beiden Module stündlich ausgeführt.

4.8.1 Password Brute Forcing CAPEC-49

Das Python Modul *capec49.py* dient zur Erkennung und Analyse des bereits vorgestellten Angriffsmusters Password Brute Forcing. Bei Aufruf per Cronjob, prüft das Modul jeweils den Zeitraum einer Stunde auf eingegangene Angriffe, die dem Angriffsmuster entsprechen.

Um den ersten Schritt des entwickelten Alarmierungsprozesses, die Erkennung des Angriffsmusters, umzusetzen, muss über eine SQL Abfrage auf die Log Datenbank des Cowrie Honeypotsensors zugegriffen werden.

```
1  #Ausschnitt aus capec49.py
2  #Abfrage des Angriffsmusters Password Brute Forcing
3  queue = "select ip, timestamp from sessions
4      join auth on sessions.id = auth.session
5      where timestamp between %s and %s
6      group by ip having count(auth.id) = 21 and
7      count(distinct auth.session) = 1  and
8      count(distinct auth.username) = 1
9      order by timestamp desc;"
10
11 cursor.execute(queue,(begin_s, end_s))
```

Die oben dargestellte SQL Abfrage prüft, ob in dem Zeitraum der letzten Stunde Sitzungen mit dem SSH Sensor eingegangen wurden, bei denen 21 Mal erfolglos versucht wurde, sich mit einem einzigen Benutzernamen am Sensor einzuloggen. Nach 21 Versuchen beendet der Cowrie Sensor automatisch die jeweilige Sitzung. Daher können keine

Sitzungen mit mehr als 21 fehlgeschlagenen Login Versuchen existieren. Sollten über diese Abfrage Angriffe identifiziert werden, werden die IP-Adresse des Angreifers und der Zeitpunkt des Angriffes aus der Cowrie Datenbank ausgelesen.

Im nächsten Schritt wird überprüft, ob für die ausgelesenen IP-Adressen bereits Einträge in der Asset Datenbank oder Observed Data Objekte auf dem Dateisystem existieren. Existiert noch kein Eintrag der IP-Adresse des Angreifers in der Asset Datenbank, wird ein neuer Data Object Datensatz für die IP-Adresse angelegt. Anschließend werden ein Observed Data Datensatz und ein zugehöriges STIX Objekt erzeugt, mit denen die IP-Adresse des Angreifers zukünftig überwacht werden kann. Zur Erzeugung der Datensätze, wird auf die Funktionen *data_object.add* beziehungsweise *observed_data.add* zurückgegriffen.

Existiert bereits ein Eintrag für die IP-Adresse des Angreifers, werden der verknüpfte Observed Data Datensatz und das zugehörige STIX Objekt, über die Funktion *observed_data.update* aktualisiert.

Um den Alarmierungsprozess abzuschließen, wird im folgenden Code Ausschnitt ein STIX Bundle aus dem Observed Data Objekt, welches die IP-Adresse des Angreifers überwacht und dem zugehörigen CAPEC Angriffsmuster, über die Funktion *stix.create_bundle* erzeugt. Anschließend wird das erzeugte Bundle über die Funktion *mail.send_mail* als IoC innerhalb einer Warnmeldung versandt.

```
1  #Ausschnitt aus capec49.py
2  #STIX Bundle erzeugen und versenden
3  stix_id = [observed_data.get_stixid(observed_id)]
4  stix_id.append("attack-pattern--7611adb7-94c4-
5                 4d2b-8326-7fe9f148ed62")
6
7  bundle = stix.create_bundle(stix_id)
8
9  #Bundle per Mail versenden
10 mail.send_mail(bundle)
```

4.8.2 Remote Code Inclusion CAPEC-253

Anders als das Python Modul *capec49.py*, verfügt das Modul *capec253.py* über ein Zusatzmodul, über welches es aufgerufen wird. Das Modul *handler.py* übernimmt für den Web-Honeypotsensor die Erkennung des jeweiligen CAPEC Angriffsmusters. Zwar wird im Rahmen der Bachelorarbeit nur ein einziges Angriffsmuster über den Web-Honeypot emuliert, allerdings bietet der Sensor die Möglichkeit, noch weitere Angriffsmuster zu emulieren. Daher wird die generierte Log-Datei innerhalb des *handler.py* Moduls, auf aufgetretene Angriffsmuster untersucht. Wird ein solches Angriffsmuster gefunden, wird das jeweilige CAPEC Modul des Web-Honeypotsensors aufgerufen.

```
1  #Ausschnitt aus handler.py
2  count = 0
3  with open('/tmp/tanner_report.json', 'rw+') as file:
4      for line in file:
5          data = json.loads(lines[count])
6
7          if(data['response_msg']['response']['message']
8          ['detection']['name'] == 'rfi'):
9              capec253.capec253(data)
10     else:
11         #Einbindung weiterer Angriffsmuster moeglich
12
13     count = count + 1
14
15     #Alle verarbeiteten Zeilen loeschen
16     file.truncate(0)
```

Dazu wird jede Zeile der json Log-Datei auf ein erkanntes Angriffsmuster überprüft. Im zuvor gezeigten Ausschnitt aus dem *handler.py* Modul, wird lediglich auf das emulierte Angriffsmuster RFI - Remote File Inclusion - geprüft. Wird dieses Muster erkannt, wird das zugehörige CAPEC Modul aufgerufen und die Daten der betreffenden Zeile übergeben. Wurden alle Zeilen der Log-Datei durchlaufen, werden die überprüften Zeilen gelöscht, damit sie beim nächsten Aufruf nicht erneut geprüft werden.

Alle weiteren Schritte des Alarmierungsprozesses werden durch die jeweiligen CAPEC Module umgesetzt.

Bevor der Alarmierungsprozess in *capec253.py* weiter abgearbeitet wird, werden Informationen über die Datei gesammelt, welche über die RFI Emulation des Honeypotsensors heruntergeladen wurde. Dazu werden über den Befehl *subprocess.check_output* zwei Kommandos auf der Linux Umgebung ausgeführt, deren Ergebnis innerhalb einer Variable gespeichert wird.

```
#Ausschnitt aus capec253.py
hash = subprocess.check_output(['find', '/opt/tanner/files/',
'-newermt' , begin,  '-not' ,'-newermt', end, '-printf', '%P'])

size = subprocess.check_output(['find', '/opt/tanner/files/',
'-newermt' , begin,  '-not' ,'-newermt', end, '-printf', '%s'])
```

Die beiden Befehle suchen über das Kommando *find* die Datei heraus, die zum Zeitpunkt des erfolgten Angriffes, auf dem Dateisystem des Servers erstellt wurde. Zusätzlich werden über die Parameter *%P*, beziehungsweise *%s* der Hash Wert, sowie die Größe der heruntergeladenen Datei ermittelt. Diese Daten werden im weiteren Ablauf benötigt, um einen entsprechenden Datensatz in der Asset Datenbank zu erzeugen.

Wie im Python Modul *capec49.py* umgesetzt, wird zunächst geprüft, ob die während des Angriffes aufgezeichneten Daten, bereits in der Asset Datenbank existieren. Im Gegensatz zu CAPEC-49, werden bei der Analyse der Log-Daten für das Angriffsmuster CAPEC-253 mehrere Daten aufgezeichnet, die das zu erzeugende Observed Data Objekt beobachtet. Dies sind die IP-Adresse des Angreifers, die heruntergeladene Datei und der HTTP-Request, der auf dem Websensor eingegangen ist.

Für jedes der drei zu beobachtenden Objekte wird geprüft, ob ein Datensatz in der Asset Datenbank vorhanden ist. Ist dies der Fall, wird die ID des zugehörigen Data Objects, über die Funktion *data_object.get_file_id* ausgelesen und gespeichert. Ansonsten wird ein neuer Eintrag in der Asset Datenbank, über die Funktion *data_object.add*, erzeugt. Während der Existenzprüfung der Daten werden jeweils zwei Prüfvariablen erstellt, die im weiteren Ablauf zur Erstellung oder Aktualisierung des Observed Data Objektes benötigt werden. Die Variable *ip_id* (exemplarische Benennung für den IP Datensatz) speichert die Data Object ID des IP Datensatzes. Die Variable *ip_check* gibt an, ob bereits ein Datensatz in der Asset Datenbank existiert.

Das zugehörige Observed Data Objekt wird erst erzeugt, wenn die Existenz aller Datensätze geprüft wurde. Es wird nur ein neues Objekt erzeugt, wenn mindestens einer der aufgezeichneten Datensätze noch nicht in der Asset Datenbank existiert. Für den Fall, dass es bereits Datensätze für alle Data Objects gibt, muss geprüft werden, ob bereits ein Observed Data Objekt existiert, welches alle drei Data Objects überwacht. Ist dies nicht der Fall, muss ebenfalls ein neues Objekt erzeugt werden.

```
#Ausschnitt aus capec253.py
if(file_check == 1 and ip_check == 1 and http_check == 1):
```

Über die Prüfvariablen wird zunächst überprüft, ob alle Datensätze bereits in der Datenbank vorhanden sind. Ist dies nicht der Fall, wird über die Funktion *observed_data.add* ein neues Observed Data Objekt erzeugt. Andernfalls werden aus der Verbindungstabelle *object_link* die jeweiligen Observed Data IDs ausgelesen, die für die einzelnen Data Object Datensätze existieren.

```
#Ausschnitt aus capec253.py
#Exemplarische Abfrage fuer den http-Request
scursor.execute("select observed_id from object_link
            where data_id = %s", (http_id,))

http_result = scursor.fetchall()
```

Anschließend wird mit Hilfe zweier geschachtelter for Schleifen geprüft, ob die ausgelesenen Oserved Data IDs übereinstimmen und somit ein Datensatz existiert, der alle drei Data Object Datensätze enthält.

```
#Ausschnitt aus capec253.py
for x in http_result:
    for y in ip_result:
        if(x[0] == y[0]):
            id_list.append(y[0])
for x in file_result:
    for y in id_list:
        if(x[0] == y):
            match = y
```

Um den Alarmierungsprozess abzuschließen, wird wie in *capec49.py* ein STIX Bundle aus den aufgezeichneten Daten erzeugt und per E-Mail versandt.

4.9 Test der Angriffsmuster und Sensoren

Für die beiden implementierten CAPEC Angriffsmuster wird jeweils getestet, ob die Automation des Alarmierungsprozesses funktioniert und bei Ausführung eines entsprechenden Angriffes eine Warnmeldung per E-Mail versandt wird.

4.9.1 Password Brute Forcing CAPEC-49

Um das Angriffsmuster Password Brute Forcing zu simulieren, wird sich per SSH mit dem Server über Port 22 verbunden. Mit Verbindung zu diesem Port wird eine Sitzung mit dem Cowrie SSH Honeypotsensor aufgebaut. Nach 21 erfolglosen Versuchen, sich mit dem Benutzernamen *root* an dem Sensor anzumelden, wird die Sitzung automatisch durch den Honeypot beendet. Anschließend geht eine E-Mail mit dem Betreff *Security Breach* von der Absendeadresse des Servers *alert@rasen.rasen-ba.de* ein, welche auf der folgenden Seite dargestellt wird.

Die ersten drei Zeilen beinhalten Parameter über die das, durch den Alarmierungsprozess erzeugte, STIX Bundle identifiziert werden kann. Ab Zeile vier beginnt das Observed Data Objekt, welches in Zeile 15 bis 17 das überwachte Data Object enthält. Die Zeilen 9 bis 13 enthalten die, durch das Pre-Processing verarbeiteten Attribute. Ab Zeile 19 wird das Attack Pattern dargestellt, welches mit dem erzeugten Observed Data Objekt in dem STIX Bundle verknüpft wurde.

```
"type": "bundle",
"id": "bundle--502e98da-e5b6-410b-9539-83aa11ae7105",
"spec_version": "2.0",
"objects": [
    {
        "type": "observed-data",
        "id": "observed-data--74610f81-5bd1-4f5e-ab60-
          7d86418a0e6c",
        "created": "2019-07-05T08:53:56.965Z",
        "modified": "2019-07-05T08:53:56.965Z"",
        "first_observed": "2019-07-05T08:53:56.965Z"",
        "last_observed": "2019-07-05T08:53:56.965Z"",
        "number_observed": 1,
        "objects": {
            "0": {
                "type": "ipv4-addr",
                "value": "84.46.52.180"}}},
    {
        "type": "attack-pattern",
        "id": "attack-pattern--7611adb7-94c4-4d2b-8326-
          7fe9f148ed62",
        "created": "2019-05-24T11:29:12.330Z",
        "modified": "2019-05-24T11:29:12.330Z",
        "name": "Password Brute Forcing",
        "external_references": [
            {
                "source_name": "capec",
                "external_id": "CAPEC-49"}]}]
```

Bei erneuter Ausführung des Vorganges, ändern sich wie gewünscht die Parameter *number_observed*, *last_observed* und *modified* des aktualisierten Observed Data Objektes. Es werden also alle Schritte des in Abbildung 3.7 dargestellten Prozesses erfolgreich bearbeitet.

4.9.2 Remote Code Inclusion CAPEC-253

Zum Testen des Remote File Inclusion Angriffsmusters, wird versucht über den HTTP-Request die Datei *exploit.txt* auf den Honeypot zu inkludieren.

```
1 www.rasen-ba.de/login.php?file=http://rasen-ba.de/exploit.txt
```

In der dadurch versendeten Warnmeldung gleicht der Aufbau der durch CAPEC-49 generierten Warnmeldung. Der Unterscheid besteht lediglich darin, dass in CAPEC-253 drei Data Object durch das erzeugte Observed Data überwacht werden. Dies sind ab Zeile 13 die IP-Adresse, ab Zeile 16 Der Network Traffic zusammen mit dem HTTP-Request und ab Zeile 29 die heruntergeladene Datei.

```
1  "type": "bundle",
2  "id": "bundle--35e3ca22-b119-4431-848f-7484764e2972",
3  "spec_version": "2.0",
4  "objects": [{
5      "type": "observed-data",
6      "id": "observed-data--7015bcf6-0f0f-43f4-b5bb-4ba96844cca0",
7      "created": "2019-07-10T06:10:03.341Z",
8      "modified": "2019-07-10T06:10:03.341Z"",
9      "first_observed": "2019-07-10T06:10:03.341Z"",
10     "last_observed": "2019-07-10T06:10:03.341Z"",
11     "number_observed": 1,
12     "objects": {
13         "0": {
14             "type": "ipv4-addr",
15             "value": "84.46.52.180"},
16         "1": {
17             "type": "network-traffic",
18             "src_ref": "0",
19             "src_port": 40922,
20             "protocols": [
21                 "tcp",
22                 "http"],
23             "extensions": {
24                 "http-request-ext": {
```

```
                    "request_method": "GET",
                    "request_value":
"/login.php?file=http://rasen-ba.de/exploit.txt"}}},

        "2": {
            "type": "file",
            "hashes": {
                "MD5": "a389cc8bf1e3fe77b61abcb29333d2ce"
                },
            "size": 12}}},
{
    "type": "attack-pattern",
    "id":"attack-pattern--1d30b31a-3381-418a-b70a-a718230851df",
    "created": "2019-07-10T06:50:32.595Z",
    "modified": "2019-07-10T06:50:32.595Z",
    "name": "Remote Code Inclusion",
    "external_references": [
    {
        "source_name": "capec",
        "external_id": "CAPEC-253"}]}]
```

Auch hier sorgt eine Wiederholung des Vorganges dafür, dass sich die Parameter *number_observed*, *last_observed* und *modified* des aktualisierten Observed Data Objektes wie gewünscht ändern.

5 Fazit und Ausblick

5.1 Fazit

Der konzipierte und implementierte Alarmierungsprozess prüft automatisiert Ereignisse, die an den eingesetzten Honeypotsensoren eingehen, auf erfolgte Angriffe. Bei einem erfolgten Angriff, werden dabei aufgezeichnete Daten in einer zentralen Instanz, der Asset Datenbank, gespeichert. Aus den gespeicherten Daten, werden Warnmeldungen generiert und per E-Mail versandt. Die in den Warnmeldungen dargestellten Daten, liegen formalisiert in einem Format zur Darstellung von Indicators of Compromise vor.

Der entwickelte Alarmierungsprozess konnte den in Kapitel 2.3 vorgestellten Correlation Process nur grundlegend implementieren. Durch den Einsatz von STIX und dem damit verbundenen Umsetzungsaufwand konnten einige Komponenten des Correlation Processes, wie in Kapitel 3.2 konzipiert, nicht eingesetzt werden.

Allerdings wurde das Ziel der Bachelorarbeit erreicht und bildet eine Grundlage, auf der weitere Arbeiten zur Analyse von CTI und Erzeugung von IoC aufbauen können.

5.2 Ausblick

Indicators of Compromise werden erzeugt, um Systeme gegen Bedrohungen zu schützen und Informationen über Bedrohungen zu verteilen. Die, durch den Alarmierungsprozess generierten, Indicators of Compromise erfüllen diesen Zweck aktuell nicht, da der Umfang der analysierten Daten bisher zu gering ist. Um einen Nutzen für die Bedrohungsanalyse und den Schutz vor Bedrohungen zu erreichen, muss der Alarmierungsprozess und das enthaltene Datenbankschema in weiterführenden Arbeiten ausgebaut werden.

Dabei können die im Rahmen der Bachelorarbeit vorgestellten Techniken und Hilfsmittel weiter verwendet werden. Es können sowohl weitere Komponenten des Correlation Processes, als auch weitere STIX Objektentitäten und CAPEC Angriffsmuster in den bestehenden Alarmierungsprozess eingebunden werden. Auch durch den Einsatz weiterer Serversysteme und somit dem Einsatz eines Honeynets, können die Analysemöglichkeiten erweitert werden.

Des Weiteren kann der in Kapitel 2.2 erwähnte Standard TAXII eingesetzt werden, um den Alarmierungsprozess zu erweitern und erzeugte IoC automatisiert zu verteilen.

Literaturverzeichnis

[1] : *CAPEC about.* – URL https://capec.mitre.org/about/index.html

[2] : *Cowrie Honeypotproject.* – URL https://github.com/cowrie/cowrie

[3] : *Cowrie Installationsanleitung.* – URL https://cowrie.readthedocs.io/en/latest/INSTALL.html

[4] *Honeypot Interaction.* https://doc.itc.rwth-aachen.de/pages/viewpage.action?pageId=6488285. – Zugegriffen: 2018-12-12

[5] : *Password Brute Forcing CAPEC-49.* – URL https://capec.mitre.org/data/definitions/49.html

[6] : *Remote File Inclusion CAPEC-253.* – URL https://capec.mitre.org/data/definitions/253.html

[7] : *SNARE Project.* – URL https://github.com/mushorg/snare

[8] : *STIX Intro.* – URL https://oasis-open.github.io/cti-documentation/stix/intro

[9] : *STIX Objektmodell Threat Actor.* – URL https://oasis-open.github.io/cti-documentation/examples/visualized-sdo-relationships

[10] : *TANNER Dorks.* – URL https://tanner.readthedocs.io/en/latest/dorks.html

[11] : *TANNER Emulatoren.* – URL https://tanner.readthedocs.io/en/latest/emulators.html

[12] : *TANNER Project.* – URL https://github.com/mushorg/tanner

[13] : *TAXII intro.* – URL https://oasis-open.github.io/cti-documentation/taxii/intro

[14] Andress, Jason: *Working with Indicators of Compromise*. – URL https://de.scribd.com/document/321717091/ISSA-Journal-May-2015

[15] Bianco, David: *Pyramid of Pain*. – URL http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

[16] Grimes, Roger A.: *Honeypots for Windows*. Apress, 2005. – URL https://books.google.de/books?id=vT2j480fAdoC. – ISBN 9781430200079

[17] Locard, E. ; Finke, W.: *Die Kriminaluntersuchung und ihre wissenschaftlichen Methoden*. Kameradschaft, 1930. – URL https://books.google.de/books?id=838rGQAACAAJ

[18] Lord, Nate: *What are Indicators of Compromise*. – URL https://digitalguardian.com/blog/what-are-indicators-compromise

[19] Spitzner, L: *Honeypots: Tracking Hackers*. Wesley, Addison, 2002. – URL http://www.it-docs.net/ddata/792.pdf. – ISBN 0-321-10895-7

[20] Valeur, Frederic ; Vigna, Giovanni ; Kruegel, Christopher ; Kemmerer, Richard: *A Comprehensive Approach to Intrusion Detection Alert Correlation*. – URL https://ieeexplore.ieee.org/document/1366134. – S. 146 - 169,

A Anhang

Auf der beigefügten CD befinden sich folgende Dateien:

- Die Bachelorarbeit als PDF Dokument
- SQL Skript *asset_db.sql* für die Erzeugung der Asset Datenbank
- Python Modul *observed_data.py*
- Python Modul *data_object.py*
- Python Modul *stix.py*
- Python Modul *mail.py*
- Python Modul *capec49.py*
- Python Modul *handler.py*
- Python Modul *capec253.py*

Glossar

Bundle Ein Bundle ist ein Objekt des STIX Formates. In einem Bundle können mehrere STIX Objekte zusammengefasst und dargestellt werden.

Data Object Data Objects sind Objekte des STIX Formates, die durch Observed Data Objekte überwacht werde. Die Data Objects bestehen aus einzelnen Datensätzen, wie z.B. IP-Adressen.

Observed Data Observed Data ist eine Objektentität des STIX Formates. Observed Data Objekte überwachen eine Anzahl von Data Objects und zeichnen die Anzahl der Beobachtungen, sowie die Zeitpunkte der Sichtungen der überwachten Data Objects auf. Im Rahmen der Bachelorarbeit werden Observed Data Objekte in Form von Datensätzen in der Asset Datenbank und als json Datei auf dem Dateisystem des Servers gespeichert.

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 Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Analyse von Angriffen auf Honeypots zur Erzeugung von Warnmeldungen und Indicators of Compromise

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