Information Security

Lure Box

Using Honeytokens for Detecting Cyberattacks

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

eingereicht von

Christoph Malin

is151514

im Rahmen des

Studienganges Information Security an der Fachhochschule St. Pölten

Betreuung

Betreuer: FH-Prof. Dr. Simon Tjoa

St. Pölten, 31. Mai 2017

(Unterschrift Verfasser) (Unterschrift Betreuer)

Ehrenwörtliche Erklärung

Ich versichere, dass

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

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

Ort, Datum *Unterschrift*

Kurzfassung

Wir befinden uns in einer Zeit des raschen Wandels, in der durch die starke Vernetzung und zunehmende Digitalisierung neue Chancen und Gefahren entstehen. Schätzungen gehen davon aus, dass allein den Unternehmen in Deutschland durch digitale Wirtschaftsspionage jährlich ein Schaden von mehr als 50 Milliarden Euro entsteht. Eine frühestmögliche Erkennung eines Angriffs ist essenziell, damit entsprechende Gegenmaßnahmen getroffen und weitere Untersuchungen eingeleitet werden können. Seit Jahrhunderten werden Köder und Fallen eingesetzt, um AngreiferInnen zu erkennen bzw. mögliche SpionInnen zu enttarnen. Dieses Prinzip kann auch in der digitalen Welt mithilfe von sogenannten “Honeytokens” angewendet werden. Ein Honeytoken ist jegliche Art von Information, die als Köder irgendwo in der physischen oder in der digitalen Welt platziert wird. Die Falle besteht darin, dass jeder Zugriff oder Gebrauch des Honeytokens ohne Wissen der AngreiferInnen überwacht wird und dadurch sofort Alarm geschlagen werden kann.

In der vorliegenden Arbeit wird die eigens entwickelte Lure Box vorgestellt. Diese ist eine auf kostenlos verfügbarer Software basierende virtuelle Maschine, in welcher alle notwendigen Komponenten vorhanden sind, um Logdaten von verschiedenen Systemen zu empfangen, zu verarbeiten, abzuspeichern und darzustellen. Zusätzlich zur Lure Box werden Möglichkeiten für den Einsatz von Honeytokens in einem Unternehmensnetzwerk beschrieben. Ein besonderer Fokus liegt dabei auf dem Einsatz im Zusammenhang mit Datenbanken. In einem umfassenden Szenario werden Honeytokens in vier verschiedenen Datenbanksystemen platziert und es wird gezeigt, wie diese konfiguriert werden müssen, um bei Zugriffen entsprechende Logdaten zu erzeugen. Die Logdaten werden dann auf die Lure Box übertragen. Diese stellt basierend auf den Logdaten fest, dass auf Honeytokens zugegriffen wurde und löst in weiterer Folge einen Alarm aus. Die Lure Box und der Einsatz von Honeytokens zeigen, dass die IT-Sicherheit in einem Unternehmen mit einfachen und doch sehr effektiven Mitteln erhöht werden kann.

Abstract

We're living in times of rapid changes in which new opportunities and threats occur due to the increasing digitalization. According to estimates, companies in Germany have a yearly loss of 50 billion dollars due to digital industrial espionage. Therefore, an early detection of an attack is essential to take action by counter-measures and further investigations. For centuries lures and traps have been used to detect attackers or to expose spies. With the help of Honeytokens this principle can also be used in the cyberspace. A Honeytoken is any kind of information which is placed as a lure in the physical or digital world. Any access to the Honeytoken is monitored and will raise a silent alarm.

This paper will present the Lure Box which was developed as part of this thesis. The Lure Box is a virtual machine based upon open source software. The box contains all necessary components to receive, process, store and view log data. In addition to the Lure Box, possible applications for Honeytokens in a company network are described. A special focus was given to the usage within databases. As part of a big scenario, Honeytokens were placed in four different database systems which were configured to log every access to the tokens. The generated log data are transferred to the Lure Box. A program running on the box can be used to trigger an alarm when logs are received, which indicate an access to a Honeytoken. The Lure Box and the usage of Honeytokens shows that IT security of a company can be increased with the help of simple but effective tools.

Inhaltsverzeichnis

1. Einleitung

Am 25. April 2017 waren auf der Webseite *haveibeenpwned.com*, auf der die Konten von gehackten Webseiten gesammelt werden, insgesamt 2,6 Milliarden BenutzerInnen-Konten von mehr als 210 AnbieterInnen hinterlegt. Alleine beim Hackangriff auf das Karrierenetzwerk LinkedIn im Jahr 2012 wurden über 100 Millionen Accounts gestohlen. [1] Interessant dabei ist, dass anfangs noch die Rede von 6,5 Millionen Konten war. [2] Im Jahr 2016, also erst vier Jahre später, stellte sich heraus, dass damals über 100 Millionen Accounts entwendet worden waren. [3] Auch bei der US-Geheimdienstbehörde NSA verging einige Zeit, bis festgestellt wurde, dass der Insider Edward Snowden zwischen 50 000 und 200 000 geheime Dokumente entwendet hatte. [4] Ihm gelang es, mit diesen nach Hongkong zu flüchten und die Welt über einen der größten Überwachungsskandale unserer Zeit aufzuklären. [5] Diese beiden Vorfälle zeigen, dass große IT-Unternehmen wie LinkedIn, aber auch vermeintlich allmächtige Geheimdienstbehörden wie die National Security Agency keine vollständige Kontrolle darüber haben, wer wann unerlaubt auf gewisse Daten zugreift.

Doch nicht nur Unternehmen und Behörden sind von Cyberkriminalität betroffen. Als Folge der zunehmenden Digitalisierung und Vernetzung schlagen Kriminelle immer häufiger auch aus Privatpersonen Kapital, indem sie sogenannte „Ransomware“ verbreiten, eine Schadsoftware, die BenutzerInnen davon abhält, ein System zu verwenden, indem es beispielsweise Dateien verschlüsselt oder den Bildschirm blockiert. Erst nach Bezahlung eines Lösegelds wird das gehackte System wieder freigegeben. [6]

Die fortschreitende Professionalisierung von Schadsoftware sowie die daraus entstandenen Schäden bringen Verantwortliche in Unternehmen unter Zugzwang. Ist es nicht der mögliche Imageschaden, der bei einem Datenverlust Angst macht, so bringen spätestens Compliance-Anforderungen oder gesetzliche Vorgaben Bewegung in die Welt der IT-Sicherheit. Ein großer Treiber sind etwa die EU-Datenschutz-Grundverordnung und Compliance-Standards wie HIPAA [7], PCI DSS [8] oder SOX [9]. Firmen nehmen bereits mehr Geld für Informationssicherheit in die Hand. Eine Studie von Garner geht davon aus, dass Unternehmen im Jahr 2016 weltweit insgesamt 86,1 Milliarden US-Dollar – und damit um 7,9 Prozent mehr als im Jahr davor – dafür ausgegeben haben. [10] Bei Verstößen gegen die EU-Datenschutz-

Grundverordnung könnten etwa Geldbußen von bis zu 20 Millionen Euro oder bis zu 4 Prozent des gesamten weltweit erzielten Jahresumsatzes des vorangegangenen Geschäftsjahres verhängt werden, je nachdem, welcher der Beträge höher ist. [11, S. 83]

Während also viel Geld in teure Software und Beratung gesteckt wird, werden die sehr günstigen und effizienten Honeytokens nur selten verwendet. Diese sind sehr stark mit den Honeypots verwandt und im Grunde nichts anderes als Köder, die an einer Stelle platziert werden, wo nur BenutzerInnen mit bösartigen Absichten auf diese stoßen können. Jeglicher Zugriff darauf bzw. jede Verwendung eines dieser Köder wird überwacht und stellt eine verdächtige bzw. verbotene Aktivität dar. So können Honeytokens beispielsweise eine Kreditkartennummer, eine Excel-Datei, ein Datenbankeintrag oder frei erfundene Zugangsdaten zu einem System sein. [12] Greift jemand darauf zu, kann Alarm geschlagen werden. Honeytokens eignen sich auch besonders gut zur Aufdeckung von Insiderattacken, die in vielen Fällen nur sehr schwer erkannt werden können. MitarbeiterInnen, die im Unternehmen arbeiten oder gearbeitet haben, versuchen dabei aus verschiedensten Motiven heraus, an bestimmte Informationen zu gelangen. Im Gegensatz zu externen AngreiferInnen kennen sich Insider mit den Abläufen im Unternehmen aus und haben vielleicht auch Zugang zu wichtigen Systemen. Im günstigsten Fall handelt es sich lediglich um Neugier, im schlimmsten Fall wird nach sensiblen und/oder vertraulichen Daten gesucht, um diese an die Konkurrenz oder interessierte Dritte zu verkaufen. Solche Diebstähle sind durch den Einsatz eines ausgeklügelten Sicherheitskonzeptes unter Einbeziehung von Honeytokens frühzeitig zu erkennen und deren negative Auswirkungen können dadurch vermindert werden. Je früher ein Angriff oder ein auffälliges Verhalten entdeckt wird, desto eher können Gegenmaßnahmen getroffen oder weitere Untersuchungen eingeleitet werden.

Da wichtige und kritische Informationen wie Daten von KundInnen oft in Datenbanken abgespeichert werden, wird im Rahmen der vorliegenden Masterarbeit ein besonderes Augenmerk auf die Platzierung und Überwachung von Honeytokens innerhalb von vier gängigen Datenbanksystemen gelegt. Deren Auditing-Funktionalitäten werden näher betrachtet, und in einem umfassenden Szenario werden die Platzierung der Tokens, die Konfiguration der Überwachung, die Verarbeitung der generierten Logdaten sowie die Alarmierung detailliert erklärt. Die Lure Box ist eine im Zuge dieser Arbeit erstellte virtuelle Maschine, welche die Logdaten kostenlos, schnell und effizient verarbeiten und abspeichern kann. Wird ein Zugriff auf Honeytokens festgestellt, so können Alarme generiert werden.

Um die Möglichkeiten von Honeytokens und deren Einsatz innerhalb von Datenbanksystemen besser

verstehen zu können, stellen sich daher folgende forschungsleitenden Fragen:

- Was wird unter Honeytokens verstanden und warum bietet sich deren Einsatz in Datenbanksystemen an, um die IT-Sicherheit zu erhöhen?
- Wie kann ein System zur Überwachung von Honeytokens in verschiedenen Datenbanken mit kostenlosen Mitteln umgesetzt werden?

1.1. Gliederung der Arbeit

Abbildung 1.1 zeigt die Gliederung dieser Arbeit. In Kapitel 2 und Kapitel 3 wird vorgestellt, worum es sich bei Honeypots – und insbesondere bei Honeytokens – handelt. Es werden deren Vor- und Nachteile beschrieben und es wird eine Zusammenfassung an verschiedenen möglichen Einsatzszenarien gezeigt. Anschließend wird in Kapitel 4 auf die Ideen hinter der Lure Box eingegangen. Es wird ein umfassendes Szenario für den Einsatz in verschiedenen Datenbanksystemen geschildert. Kapitel 5 vermittelt notwendiges Hintergrundwissen, um der Umsetzung der Szenarien ab Kapitel 6 besser folgen zu können.

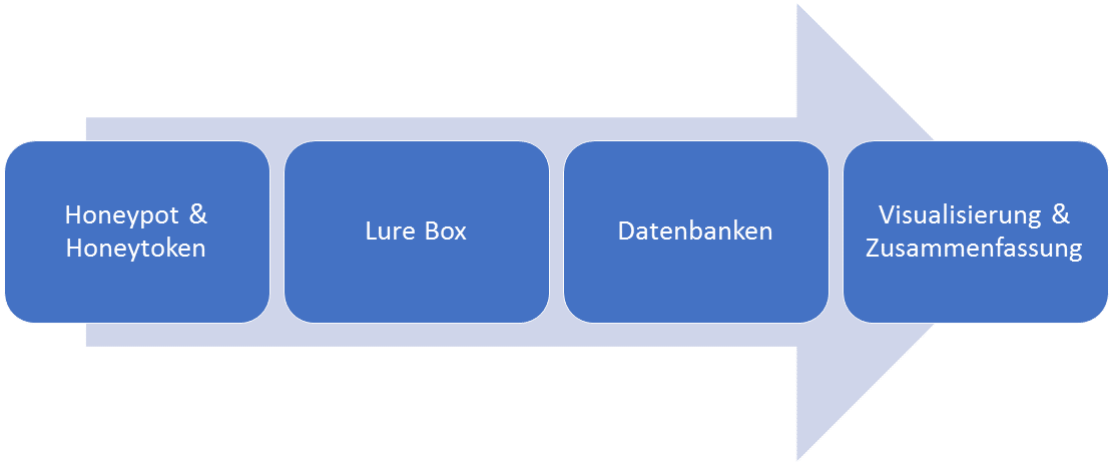

Abbildung 1.1.: Die Gliederung dieser Arbeit.

Kapitel 7, Kapitel 8, Kapitel 9 und Kapitel 10 zeigen, wie in den verschiedenen Datenbanken Honeytokens platziert und auditiert werden. Kapitel 11 erklärt, wie ein Postfix E-Mail Server eingesetzt wird, um die Verwendung von Honeytoken-E-Mail-Adressen zu erkennen. Am Ende der Arbeit zeigt Kapitel 12, wie ein generierter Alarm aussehen kann und wie die gesammelten Logdaten mittels Kibana angesehen werden können. Abschließend gibt Kapitel 13 eine Zusammenfassung der Arbeit und einen Ausblick für zukünftige Arbeiten.

2. Honeypot

Das folgende Kapitel erklärt, was Honeypots sind und wo diese platziert werden können. Es werden die Vor- und Nachteile des Einsatzes von Honeypots abgewogen und es wird auf die Unterschiede zu Honeynetzen und Honeytokens eingegangen.

2.1. Definition

„Honeypots“ oder „Honeynetze“ sind Systeme, die vermeintliche Schwachstellen enthalten und an Orten im Netzwerk platziert werden, an denen AngreiferInnen danach suchen. Mit ihrer Hilfe kann die Sicherheitsabteilung eines Unternehmens Informationen über AngreiferInnen sammeln. Über die Vorgehensweise der AngreiferInnen lassen sich deren eingesetzte Tools sehen und gegebenenfalls auch Rückschlüsse auf das Wissen einer Angreiferin bzw. eines Angreifers ziehen. [13, S. 163]

Honeypots sind besonders gut zum Erkennen einer Bedrohung durch InsiderInnen geeignet. Diese sind aktuelle oder ehemalige MitarbeiterInnen, LieferantInnen oder andere UnternehmenspartnerInnen, die Zugriff auf Netzwerke, Systeme und Daten im Unternehmen haben bzw. hatten. [14, S. 1] Es gibt viele Faktoren, die MitarbeiterInnen dazu bringen können, zu InsiderInnen zu werden, darunter Vergeltung/Rache, finanzielle Vorteile oder Unzufriedenheit. [15, S. 25–26] Die Besonderheit in solchen Fällen ist, dass diese Personen mit den Abläufen im Unternehmen vertraut sind, über legitime Berechtigungen verfügen und ihre Angriffe daher gezielt planen können.

> „Honeypots werden eingesetzt, um von anderen Systemen abzulenken, neue Angriffsmethoden zu identifizieren oder automatisch Gegenmaßnahmen zu ergreifen.“ [16]

Generelles Merkmal eines Honeypots ist, dass kein System jemals darauf zugreifen sollte. [17, S. 59] Des Weiteren hat ein Honeypot keinen Wert für die Produktion in einem Unternehmen. Per definitionem sollten also im Normalfall keine Aktivitäten auf einem Honeypot stattfinden, weshalb jegliche Kommunikation mit dem Honeypot als Anomalie angesehen werden kann. [18, S. 171] Der Wert eines Honeypots

wird generell dadurch bestimmt, von welchen Bedrohungen er genutzt wird. Findet auf einen bestimmten Honeypot keinerlei Zugriff statt, so hat dieser nur einen geringen Wert. [18, S. 170]

Honeypots lassen sich nach der Art der Interaktion und nach der Art der Implementierung einteilen: [13, S. 165], [19]

- **Art der Interaktion**
  - **Honeypots mit niedriger Interaktion** simulieren nur einen Dienst wie zum Beispiel einen HTTP-Server oder einen FTP-Server. Sie werden so gebaut, dass sie von AngreiferInnen nicht übernommen werden können. Sobald ein Zugriffsversuch auf das System stattfindet, wird ein Alarm generiert. Beispiele dafür sind etwa der Dionaea Honeypot [20], der Honeyd Honeypot und [21] Conpot [22].
  - **Honeypots mit hoher Interaktion** simulieren ein vollständiges Betriebssystem und können von AngreiferInnen vollständig übernommen werden, was diesen erlauben würde, zusätzliche Software zu installieren und damit weitere Attacken zu starten. Der Honeypot zeichnet alle Aktivitäten des Angriffs auf; dadurch können Rückschlüsse auf die Fähigkeiten der angreifenden Person geschlossen werden.
- **Art der Implementierung**
  - **physisch:** eine physische Maschine im Netzwerk mit einer eigenen IP-Adresse
  - **virtuell:** wird von einer anderen Maschine simuliert und antwortet auf Netzwerkverkehr, der an den virtuellen Honeypot gesendet wird

Eine umfassende (von Jose Nazario erstellte) Liste von verfügbaren Honeypots kann auf *github.com* eingesehen werden. [23]

Zwei besondere Formen von Honeypots sind „Honeynetze“ und „Honeytokens“. Honeynetze sind komplette Netzwerke von Computern, die nicht emuliert sind, daher können in einem Honeynetz ähnliche Systeme wie in der produktiven Umgebung eines Unternehmens platziert werden. Innerhalb dieser Systeme lassen sich zusätzliche Informationen wie Dateien, Datenbankeinträge, Logeinträge und andere Informationen platzieren, an denen potenzielle AngreiferInnen interessiert sein könnten. Diese Einträge werden „Honeytokens“ genannt. [18, S. 172] Weitere Informationen über Honeytokens finden sich in Kapitel 3.

2.2. Vor- und Nachteile

Der große Vorteil von Honeypots besteht in den geringen Kosten, die mit ihrer Installation verbunden sind. Da Honeypots darüber hinaus einen wichtigen Beitrag bei der Sammlung von Informationen über einen Angriff leisten, können Rückschlüsse auf die Fähigkeiten und Ziele von AngreiferInnen gezogen werden. Weiters können Honeypots von Produktivsystemen ablenken und die Erkennung von InsiderInnenattacken unterstützen. [18, S. 171]

Bei all den genannten Vorteilen kann deren Einsatz jedoch auch problematisch sein. Honeypots laufen oftmals auf Systemen, die veraltet sind und daher Schwachstellen aufweisen. Neben den bewusst simulierten Schwachstellen vermögen AngreiferInnen also auch die unbeabsichtigten Schwachstellen auszunutzen, um davon ausgehend weitere Angriffe auf das Netzwerk zu starten. Aus diesem Grund ist es wichtig, dass Honeypots und die darunterliegenden Systeme kontinuierlich auf dem aktuellen Stand gehalten und überwacht werden. [13, S. 164] Ein weiteres Problem tritt auf, wenn ein Eindringling erkennt, dass er/sie es mit einem Honeypot zu tun hat. In diesem Fall könnte der Honeypot dazu benutzt werden, um das Sicherheitsteam eines Unternehmens mittels vorgespielten bzw. gefälschten Angriffen auf den Honeypot abzulenken. [24] Auch gesetzliche Regelungen können das Betreiben eines Honeypots erschweren. [25, S. 397]

Es darf nie vergessen werden, dass ein Honeypot immer nur das mitschneiden kann, was direkt mit ihm interagiert wird. Auch hat er keinen Wert, wenn niemand auf ihn zugreift. In jedem Fall ist es essenziell, dass ein Honeypot niemals innerhalb eines Netzwerkes platziert werden sollte, ohne Teil eines größeren Sicherheitskonzeptes zu sein.

2.3. Platzierung von Honeypots

Der Honeypot muss an einer Stelle platziert werden, wo er attraktiv auf AngreiferInnen wirkt, und es muss die Möglichkeit bestehen, vom Honeypot aus Logdaten an ein Sicherheitsteam zu übermitteln. Folglich kann ein Dateiserver, der im Netzwerk für alle UserInnen verfügbar ist, diesen Zweck nicht erfüllen, da auch gewöhnliche MitarbeiterInnen darauf nach Dateien suchen könnten. [13, S. 164]

2.3.1. Platzierung in einer DMZ

In einer demilitarisierten Zone (DMZ) müssen Honeypots wohlüberlegt platziert und konfiguriert werden, da das Internet permanent gescannt wird und automatisierte Tools laufend Attacken auf Webserver

und Mailserver durchführen. [13, S. 163] So könnte ein Webserver-Honeypot in der DMZ platziert werden. Der Webserver wird nicht produktiv genutzt und es wird ihm auch kein DNS-Eintrag zugewiesen; er befindet sich lediglich im gleichen Netzwerk wie die anderen Webserver. Findet von einem produktiven Webserver eine Verbindung in Richtung Honeypot statt, so kann davon ausgegangen werden, dass der produktive Server kompromittiert wurde. [18, S. 171]

2.3.2. Platzierung in einem internen Netzwerk

Während bei einem Honeypot in einer DMZ davon ausgegangen werden kann, dass dieser laufend angegriffen wird, ist dies im internen Netzwerk eher die Ausnahme. Wird auf einen Honeypot im internen Netzwerk zugegriffen, so haben es die AngreiferInnen bereits an der äußeren Firewall vorbeigeschafft oder es handelt sich um InsiderInnen im Unternehmen. [13, S. 164] Daraufhin werden die betreffenden Personen vermutlich damit beginnen, das Netzwerk weiter zu erkunden, um ihre Mission abschließen zu können. Die richtige Platzierung des Köders innerhalb des Netzwerkes ist daher eine zentrale Herausforderung, die wohlüberlegt gelöst werden muss. [13, S. 164] Zur Ermittlung eines geeigneten Standortes können folgende Fragen helfen: [13, S. 165]

1. Welche sind die wertvollsten Assets im Unternehmen?
2. Welche Systeme beinhalten diese Assets?
3. Wie würden AngreiferInnen versuchen, auf diese Systeme zuzugreifen?

3. Honeytokens

Das nachfolgende Kapitel bietet eine Definition zu Honeytokens und geht auch auf deren Geschichte ein. Erläutert werden die Vor- und Nachteile sowie jene Aspekte, auf die bei der Erzeugung und Platzierung geachtet werden muss. Eine Reihe von verfügbaren Tools im Zusammenhang mit Honeytokens wird vorgestellt und eine Vielzahl an möglichen Einsatzszenarien wird zusammengefasst.

3.1. Definition

Der Begriff „Honeytoken“ wurde erstmals im Jahr 2003 von Augusto Paes de Barros in der Honeypot Mailing List von Security Focus verwendet. [26] Das Konzept des Honeytokens wurde anschließend von Lance Spitzner, dem Moderator der Mailing-Liste, aufgegriffen und er veröffentlichte den Artikel *Honeytokens: The Other Honeypot* [12], durch den der Begriff „Honeytoken“ öffentlich bekannt gemacht wurde und der es der Information Security Community ermöglichte, darüber zu diskutieren. [24]

Honeytokens gibt es in den verschiedensten Formen und Größen, aber ihnen allen liegt das gleiche Konzept zugrunde: Er ist eine digitale oder analoge Ressource, deren Wert darin liegt, von jemandem unautorisiert verwendet zu werden. [12] Wie bei einem Honeypot sollten im Normalfall auch beim Honeytoken keine legitimen BenutzerInnen darauf zugreifen oder diese verwenden. [27, S. 1313] Der Unterschied des Honeytokens zu einem Honeypot liegt darin, dass ein Honeytoken niemals ein Computer ist. [12]

Shabtai et al. stellten eine Unterteilung in aktive und passive Honeytokens an. Aktive Honeytokens sind Fake-Einträge, die bewusst unter echten Daten platziert werden. Jegliche Interaktion mit aktiven Honeytokens ist grundsätzlich legitim. Erst bei einer missbräuchlichen Verwendung der Daten wird Alarm geschlagen. Gibt es beispielsweise eine Datenbank mit Telefonnummern von KundInnen, können darin Fake-KundInnen mit echten Telefonnummern angelegt werden. Wird eine dieser erfundenen Personen von jemand anderem als den VerkäuferInnen angerufen, so kann davon ausgegangen werden, dass Daten entwendet wurden. [28, S. 9:2] Passive Honeytokens hingegen werden an Orten platziert, auf die BenutzerInnen normalerweise niemals zugreifen würden. Daher ist jedwede Interaktion mit einem passiven

Honeytoken nicht legitim und es kann Alarm geschlagen werden. [28, S. 9:2]

Honeytokens sind in der Literatur und im Internet häufig auch unter den Begriffen „Canarytoken“, „Decoys“ und „Lures“ zu finden. [29, S. 242]

3.2. Geschichte

Während der Einsatz von Ködern und Fallen schon seit Jahrhunderten Teil der Kriegsführung ist [30], ist der erste generelle Einsatz von Honeytokens für das Jahr 1987 dokumentiert. [31] Im Buch *Kuckucksei* beschreibt Clifford Stoll, wie es ihm als Systemadministrator am Lawrence Berkeley National Laboratory (LBL) gelang, bei der Ergreifung des deutschen Hackers Markus Hess einen wesentlichen Beitrag zu leisten. [31] Stoll entdeckte nicht autorisierte Zugriffe auf das LBL und stellte fest, dass sich ein Hacker aus Europa persistent mit dem Netzwerk des LBL verbunden hatte und nach Begriffen wie *stealth* und *nuclear* suchte.

Weil man für eine Standortbestimmung von Verbindungen aus Europa zu jener Zeit eine Stunde benötigte, musste ein Weg gefunden werden, um den Hacker dazu zu bringen, die Verbindung für eine ganze Stunde aufrechtzuerhalten. Stoll schlug also vor, eine größere Menge von fiktiven Informationen zu platzieren, die den Hacker dazu verleiten sollten, länger als die für die Standortbestimmung benötigte Zeit mit dem LBL verbunden zu bleiben. Die fiktiven Informationen, die Stoll für den Angreifer erstellte, wurden in einem Account mit dem Namen *SDINET* gespeichert und sollten wie Details zu einem neuen Militärprojekt aussehen. Sämtliche Dateien waren mit einem Alarm versehen, der ausgelöst wurde, wenn auf diese ein Zugriff stattfand. Es gab keine legitimen BenutzerInnen, die einen Grund gehabt hätten, auf diese Dateien zuzugreifen. Eines dieser Dokumente beinhaltete zusätzlich ein Formular zur Anforderung von geheimen Informationen zu dem Projekt sowie eine Adresse, wo die geheimen Daten angefordert werden könnten. Am 16.01.1987, eine Woche nachdem die Honeytokens platziert worden waren, konnte Stoll mithilfe der Deutschen Bundespost die Adresse des Angreifers ermitteln. [24] Am 27.04.1987 erreichte das Formular für die Anforderung der geheimen Informationen eine vom Federal Bureau of Investigation (FBI) kontrollierte Adresse. Über die Rücksendeadresse konnte das FBI einen Komplizen festnehmen. [24]

Auch Unternehmen, die Karten und Pläne von Gegenden erzeugen, wendeten dieses Prinzip an. Sie platzierten Straßen und sogar ganze Orte und Städte, die es in Wirklichkeit gar nicht gab, um zu erkennen, ob die eigenen Karten von der Konkurrenz kopiert wurden. [24] AutorInnen von Enzyklopädien platzierten

frei erfundene Begriffe in ihren Werken, um beweisen zu können, dass die Konkurrenz die Informationen lediglich kopiert hatte. [24]

Die „John F. Kennedy“-Krankenakte ist ein weiteres klassisches Beispiel für den Einsatz eines Honeytokens. In Krankenhäusern ist es den MitarbeiterInnen oftmals untersagt, beliebige PatientInnendaten einzusehen. Gibt es MitarbeiterInnen, die trotzdem Einsicht in Akten nehmen, obwohl es dafür keinen plausiblen Grund gibt, so möchte die Krankenhausleitung aus verständlichen Gründen darüber informiert werden. Bei dem besagten Beispiel wird ein gefälschter Krankenbericht mit dem Namen „John F. Kennedy“ erstellt und in der Datenbank angelegt. Dieser Bericht dient als Honeytoken. Gibt es neugierige MitarbeiterInnen, so wird der Name des ehemaligen Präsidenten der USA hervorstechen und die neugierige Mitarbeiterin bzw. der neugierige Mitarbeiter wird vermutlich darauf zugreifen. Sobald ein Zugriff erfolgt, wird ein Alarm generiert. [12]

3.3. Vor- und Nachteile

Honeytokens sind sehr günstig, da sie sehr schnell erzeugt werden können. Es muss keine spezielle Technologie installiert und kein Hersteller kontaktiert werden, und es sind auch keine Lizenzgebühren zu bezahlen. [12] Ein weiterer großer Vorteil besteht in der hohen Flexibilität. Da Honeytokens sehr einfach sind, können sie in vielen unterschiedlichen Systemen platziert werden. So kann beispielsweise eine Honeytoken-E-Mail-Adresse in einer KundInnenkartei abgelegt werden. Wird zu dieser Adresse nun von einer anderen Stelle als von der Verkaufsabteilung Kontakt aufgenommen, so weiß man, dass die Daten missbräuchlich verwendet wurden. AngreiferInnen verraten sich durch die Verwendung im Grunde selbst. Ein weiterer Vorteil ist, dass je nach Implementierung sogar verschlüsselte Angriffe erkannt werden können. Für manche Honeytokens spielt es keine Rolle, wie ein Angriff durchgeführt wird. Es wird lediglich der Zugriff darauf festgestellt. [32]

Der größte Nachteil von Honeytokens liegt darin, dass diese einen Angriff nur dann erkennen können, wenn während des Angriffs auf den Honeytoken auch tatsächlich zugegriffen wird. Werden lediglich andere Teile des Systems attackiert, so wird kein Alarm ausgelöst. Aus diesem Grund ist es umso wichtiger, Honeytokens als Teil eines größeren Sicherheitskonzeptes zu sehen und niemals auf Honeytokens alleine zu vertrauen. Je nachdem, wie die Überwachung des Zugriffs auf einen Honeytoken implementiert wird, könnten AngreiferInnen beispielsweise Honeytokens vor der Übertragung komprimieren oder verschlüsseln. Wird dann die Überwachung mittels eines Intrusion Detection System (IDS) durchgeführt, würde der Zugriff nicht erkannt werden. [32]

3.4. Anforderungen an die Generierung und Platzierung von Honeytokens

Ein zentraler Aspekt beim Einsatz von Honeytokens ist, die Daten so attraktiv aussehen zu lassen, dass AngreiferInnen nicht widerstehen können, darauf zuzugreifen. Findet ein Zugriff statt, so müssen die betreffenden Personen aber auch jene Inhalte vorfinden, die von ihnen erwartet werden. Eine Datei mit dem Namen *Kreditkartennummern.xlsx* sollte beispielsweise auch wirklich Zahlenkombinationen enthalten, die wie echte Kreditkartennummern aussehen. [13, S. 166]

Bowen et al. definierten mehrere Eigenschaften, die beim Designen und Verteilen von glaubwürdigen Ködern/Decoys zu beachten sind. Diese wurden für Honeytokens leicht angepasst übernommen: [33]

1. **Glaubwürdigkeit** – AngreiferInnen müssen davon überzeugt sein, dass es genau das ist, wofür es sich ausgibt.
2. **Verlockend** – Es können nur Angriffe erkannt werden, wenn auf den Honeytoken zugegriffen wird. Daher sollte dieser sehr attraktiv auf AngreiferInnen wirken.
3. **Hervorstechend** – AngreiferInnen müssen den Honeytoken gut finden können.
4. **Erkennbarkeit** – Der Honeytoken ist nur etwas wert, wenn Zugriffe darauf erkannt werden.
5. **Variabilität** – AngreiferInnen sollten Honeytokens nicht als solche erkennen können, wenn sie ein paar Eigenschaften mit echten Daten teilen.
6. **Keine Beeinflussung** – Die Honeytokens sollten Aktivitäten von legitimen BenutzerInnen nicht beeinflussen.
7. **Haltbarkeit** – Honeytokens haben eine begrenzte Haltbarkeit, nach deren Ablauf sie ausgetauscht werden sollten.

Bei der Generierung von Honeytokens sollten die oben genannten Anforderungen eingehalten werden. Ein guter Honeytoken ist einer, der selbst von ExpertInnen in dem jeweiligen Fachgebiet nicht als solcher von echten Dokumenten unterschieden werden kann. [34, S. 78] Wichtig ist es auch, diese so zu platzieren, dass nicht Hunderte Alarme täglich generiert werden, da diese ansonsten mit großer Wahrscheinlichkeit nach einiger Zeit ignoriert werden und dann keinen effektiven Nutzen mehr haben.

3.5. Projekte und Tools

Im Zuge der Recherchen für diese Arbeit konnten verschiedene Projekte und Tools entdeckt werden, die sich mit Honeytokens beschäftigen:

- **HoneyGen** – Ein von Bercovitch et al. entwickeltes Tool für die automatische Generierung von Honeytokens. Das Programm erzeugt die Tokens basierend auf den Charakteristika und Eigenschaften von bereits bestehenden Daten. Sind die Honeytokens generiert, wird eine Reihe von ähnlichen Tests durchgeführt, um zu evaluieren, ob eine Person in der Lage wäre, den generierten Token als Honeytoken zu erkennen. [35, S. 131]
- **ADHD (Active Defense Harbinger Distribution)** – Eine auf Ubuntu basierende Distribution, die eine Vielzahl von Tools zur Generierung von Honeypots und Honeytokens enthält. [36], [37]
- **Honeybits** – Ein Tool zur Generierung von Spuren, welche die AngreiferInnen zu Honeypots oder Honeytokens führen sollen. So werden beispielsweise in die *bash_history* interessante Kommandos eingefügt. Wird auf die darin verwendeten Dateien zugegriffen, kann Alarm geschlagen werden. [38]
- **Canarytoken** – Ein Webservice, bei dem Honeytokens online generiert und dann in Dokumenten platziert werden. Greift jemand auf die Honeytokens zu, wird eine E-Mail-Benachrichtigung von den BetreiberInnen der Webseite versendet. Das Projekt unterstützt unter anderem URL-Tokens, DNS-Tokens, E-Mail-Tokens, Microsoft Word-Dokumente, Adobe Reader PDF-Dokumente, SQL Server-Datenbanken, Webseiten und QR-Codes. [39], [40], [41]
- **DCEPT (Domain Controller Enticing Password Tripwire)** – Ein von der Firma SecureWorks entwickeltes Tool, das Honey Credentials in den Arbeitsspeicher von Windows-Computern legt. Wird am Active Directory Domain Controller ein Kerberos-Paket empfangen, das die Honey Credentials beinhaltet, kann Alarm geschlagen werden. [42], [43]

3.6. Erkennungsmöglichkeiten

Der beste Honeytoken ist wertlos, wenn nicht erkannt werden kann, dass auf diesen zugegriffen wurde. Für das Erkennen von Zugriffen bieten sich unter anderem die folgenden Möglichkeiten an:

- **Zugriff auf den Honeytoken** – Greift eine Person auf den Honeytoken zu, so schlägt das System, in dem dieser gespeichert ist, Alarm und es können erste Untersuchungen gestartet werden. [44,

S. 8] Der Nachteil dieser Methode ist, dass AngreiferInnen das komplette System übernommen haben könnten, das den Alarm versendet. Dann wären sie gewarnt.

- **Zugriff auf den Inhalt des Honeytokens** – Verwenden AngreiferInnen Daten wie zum Beispiel Zugangsdaten zu Servern, Telefonnummern, E-Mail-Adressen oder Kreditkartennummern, die im Honeytoken hinterlegt sind, so kann Alarm geschlagen werden. [44, S. 8]

- **Beacons** – Ein Beacon, zu Deutsch „Signalfeuer“, ist ein in Dateien eingebauter Mechanismus. Wird die betreffende Datei geöffnet oder verwendet, so wird jedes Mal ein Signal zu einem zuvor definierten Server gesendet. Beacons sind die einzige Möglichkeit, die Verwendung einer Datei zu erkennen, wenn diese das Unternehmen bereits verlassen hat. Ein Nachteil besteht darin, dass die AngreiferInnen erkennen könnten, dass die Datei versucht, ein Signal zu versenden, und dadurch Verdacht schöpfen könnten. [44, S. 8]

- **Intrusion Detection System (IDS)** – Ein IDS könnte den Netzwerkverkehr des Unternehmens überwachen. Wird die Übertragung eines Honeytokens erkannt, so kann Alarm geschlagen werden. Der große Vorteil des IDS ist, dass AngreiferInnen nur sehr schwer erkennen können, ob der Netzwerkverkehr an bestimmten Punkten überwacht wird. Ein Nachteil ist, dass die Honeytokens vor der Übertragung verschlüsselt oder komprimiert werden könnten; in diesen Fällen wäre keine Erkennung mehr möglich. [12]

3.7. Anwendungsszenarien

Der folgende Abschnitt setzt sich aus möglichen Anwendungsszenarien zusammen, die aus verschiedensten Quellen stammen und mit eigenen Ideen ergänzt wurden. Die Szenarien wurden entsprechend den bereits von Zohar et al. vorgeschlagenen Kategorien in dateibasierte, netzwerkbasierte, applikationsbasierte und zugangsdatenbasierte Szenarien eingeteilt. [44, S. 12] Die Definition, welche Szenarien einer bestimmten Kategorie zugeordnet werden, wurde selbst erstellt.

3.7.1. Dateibasierte Szenarien

Die nachfolgenden Szenarien basieren auf der Annahme, dass AngreiferInnen direkten Zugriff auf Dateien in einem Dateisystem haben.

- **Konfigurationsdateien** – Eine Textdatei, die vorgibt, die Konfigurationsdatei eines Programms zu sein. Darin werden auch Zugangsdaten abgelegt. [44, S. 13]

- **Technische Dokumente** – Dateien mit Informationen über den Zugriff auf sowie den Gebrauch von anderen Systemen, wie Passwortdateien (*passwort.txt*), BenutzerInnen-Handbücher und Sicherheitsdokumente wie *VPN-Anleitung.docx*. [45, S. 118]
- **Persönliche Dokumente** – Persönliche Dokumente von MitarbeiterInnen könnten von den AngreiferInnen bei einem Social-Engineering-Angriff zwecks Erpressung benutzt werden. [44, S. 13]
- **IT-/Unternehmensdokumente** – Kreditkartennummern, geistiges Eigentum oder erwartete Börsenkurse bieten sich an. [45, S. 118]
- **Logdateien** wie zum Beispiel die Historie von Eingaben in eine Shell oder weil Logdateien Spuren eines Angriffs beinhalten könnten. [44, S. 13], [45, S. 118]
- **Berechtigungen im Dateisystem** – Die Verwendung der Berechtigungen im Dateisystem hat zweierlei Vorteile. Zum einen können die Honeytokens für legitime BenutzerInnen unerreichbar aufbewahrt werden, zum anderen werden AngreiferInnen sehr gerne auf Ordner zugreifen, die Bezeichnungen wie „Geheim“ oder „Forschung“ aufweisen, wenn dafür spezielle Berechtigungen benötigt werden. [44, S. 18]
- **Kürzlich verwendete Dateiliste** – Von Windows, MS Office und anderen Programmen. Darin kann ein Köder platziert werden. [44, S. 13]

3.7.2. Netzwerkbasierte Szenarien

Die nachfolgenden Szenarien basieren auf der Annahme, dass die AngreiferInnen den Netzwerkverkehr abhören oder auf einen im Netzwerk zugänglichen Service zugreifen.

- **Webserver** – Auf öffentlich und intern verfügbaren Webseiten können Honeytokens platziert werden. Besonders bei ausschließlich intern verfügbaren Webseiten bieten sich zahlreiche Möglichkeiten an, da diese in der Regel nicht täglich von verschiedenen Personen attackiert werden.
  - **HTML-Kommentare** – Im Quelltext einer Webseite könnten vermeintlich vergessene Kommentare platziert werden, die auf andere Systeme zeigen oder Zugangsdaten beinhalten. [46, S. 92]
  - **Datei robots.txt** – Die *robots.txt* ist eine kleine Textdatei, die im Wurzelverzeichnis eines Webservers abgelegt wird. Suchmaschinen wie Google verwenden solche Dateien, um zu erfahren, auf welche Seiten auf einem Webserver nicht zugegriffen werden soll. Daher zählen

diese Dateien für AngreiferInnen und automatisierte Tools zu den ersten Orten, wo nachgesehen wird. Es könnte etwa auf Verzeichnisse wie */admin* oder */login* verwiesen werden. Findet darauf ein Zugriff statt, so kann Alarm geschlagen werden. [46, S. 92]

- **Platzierung von unsichtbaren Links** – Es könnten weiße Links auf weißem Hintergrund an zufälligen Stellen auf einer Webseite platziert werden. Legitime BesucherInnen werden nicht auf diese Links stoßen, sehr wohl aber Programme, die Webseiten herunterladen und diese nach Links absuchen. Eine Anfrage an den unsichtbaren Link würde einen Alarm auslösen. [46, S. 92–93]
- **Honeytoken-URL** – Für Webseiten könnten URLs wie `www.example.com/index.php?ArticleID=1234&Admin=false` verwendet werden. Wird der Admin-Parameter mittels *true* aufgerufen, so kann Alarm geschlagen werden. [27, S. 1314]
- **Automatisierte Tools** – Bei der Suche nach Schwachstellen werden oftmals automatisierte Tools eingesetzt, die einen Webserver nach bestimmten Verzeichnissen absuchen. Wird solch eine URL aufgerufen, so kann Alarm geschlagen werden. [27, S. 1314]
- **Soziale Netzwerke** – In einem sozialen Netzwerk könnte ein Fake-Account erstellt werden, der viele echte „FreundInnen“ aus dem Unternehmen hat. Einmal eingerichtet, sollte dieser keine Nachrichten von den echten MitarbeiterInnen erhalten. Wird dennoch eine Botschaft von MitarbeiterInnen aus dem Unternehmen empfangen, so kann davon ausgegangen werden, dass jener Account gehackt wurde. Der Fake-Account sollte bevorzugt aus dem Bereich Human Resources (HR) oder Finance stammen. [46, S. 93]

- **E-Mail-Server**
  - **E-Mail-Empfang** – Werden unter einer bestimmten E-Mail-Adresse, die keine externe Person kennen sollte, Nachrichten empfangen, so kann davon ausgegangen werden, dass Daten von einem bestimmten Ort exfiltriert wurden.
  - **AbsenderInnen von Phishing-Nachrichten enttarnen** – Wird ein Phishing-Versuch entdeckt, so kann auf die Nachricht geantwortet werden, indem beispielsweise Zugangsdaten zu einem System eingegeben werden, das in Wahrheit ein Honeypot ist. Beim Honeypot kann dann die Kombination aus Browser, Java, Flash und IP-Adresse etwas über die AngreiferInnen verraten.
  - **Unautorisierten Zugriff auf E-Mails enttarnen** – Neaves schlug eine Methode vor, um unautorisierte Zugriffe auf E-Mails festzustellen. Ein 1 x 1 Pixel großes Bild wird in eine

HTML-E-Mail inkludiert. Wird diese E-Mail geöffnet und werden die darin verlinkten externen Bilder geladen, so wird bei dem Server, auf dem das Bild gehostet wird, ein Logeintrag erzeugt. Mittels diesem kann festgestellt werden, welche Adresse darauf zugegriffen hat. [47] Als Alternative dazu könnte einfach ein in einer E-Mail zu erwartendes Bild, wie z. B. das Logo eines Unternehmens, platziert werden. Dieses könnte mit dem richtigen Logo identisch sein, außer dass als URL eine Adresse verwendet wird, die für Honeytokens vorgesehen ist.

Auch über die Verwendung von verlockenden Inhalten in einer E-Mail könnten neugierige MitarbeiterInnen enttarnt werden. Es könnte eine E-Mail, wie in Listing 3.1 zu sehen, im Posteingang eines Mitglieds des Managements platziert werden.

```
1  An: Finanzchef
2  Von: Help Desk
3  Betreff: Zugang zur Finanzdatenbank
4  Guten Tag,
5
6  der von Ihnen angeforderte Zugang wurde wie gewünscht angelegt. Ihre neuen
     ↪ Zugangsdaten können Sie nachstehend finden. Benötigen Sie irgendwelche
     ↪ Hilfe oder Unterstützung, so zögern Sie nicht, uns zu kontaktieren.
7
8  https://finanzen.fhstp.ac.at
9  Benutzername: cfo
10 Passwort: FH!IST!TOLL
```

Listing 3.1: Platzierung einer E-Mail-Nachricht im Posteingang eines Managementmitglieds in einem Unternehmen. Diese Nachricht dient zur Enttarnung von neugierigen MitarbeiterInnen sowie von AngreiferInnen.

Durchsuchen AngreiferInnen die E-Mails des CFOs, so werden sie auf diese interessante Nachricht stoßen. Findet dann ein Zugriff auf das System `https://finanzen.fhstp.ac.at` statt, kann Alarm geschlagen und der CFO dahingehend informiert werden, dass seine E-Mails gelesen wurden.

- **DNS-Server**
  - **Fake-Einträge** – Es gibt Tools, die mittels vorgefertigter Listen DNS-Server abfragen, ob gewisse Subdomains wie z. B. *admin.*, *api.*, *webmail.* o. dgl. existieren. Manche dieser Domains könnten als Fake-Einträge am DNS-Server angelegt werden. Sobald solche Domains abgerufen werden, kann ein Alarm generiert werden. [46, S. 93]

- **Datenbankserver**
  - **Inhalte von Tabellen** – In den Tabellen einer Datenbank könnten Telefonnummern, E-Mail-Adressen oder Kreditkartennummern hinterlegt werden. Wird auf diese zugegriffen oder werden diese verwendet, so kann Alarm geschlagen werden. Čenys et al. erzeugten Fake-Tabellen, die interessante Namen wie „Kreditkartentabelle“ hatten, und platzierten diese in echten Datenbanken, um bösartige InsiderInnenaktivitäten zu erkennen. [48]
  - **Honey-KundInnen** – Ein Unternehmen beauftragt ein Marketingunternehmen damit, Werbung für ein Produkt bei ihren KundInnen über deren E-Mail-Adressen zu machen. Gemäß dem Vertrag darf die beauftragte Marketingfirma die KundInnen ausschließlich mit der Werbung kontaktieren, die gemeinsam beschlossen wurde. Um die Einhaltung des Vertrages zu überwachen, können sogenannte Honey-KundInnen in der Datenbank platziert werden. Kommen bei diesen auch andere Werbe-E-Mails an, als ausgemacht wurde, weiß die Firma, dass der Vertrag nicht eingehalten wurde. [35, S. 132]
  - **Honey-KundInnen pro MitarbeiterIn** – Eine Universität besitzt eine Datenbank, in der die Noten und diverse persönliche Informationen der StudentInnen gespeichert sind. Auf die Datenbank haben verschiedene MitarbeiterInnen Zugriff. Honeytokens können so in der Datenbank platziert werden, dass jeder Zugriff auf einen bestimmten Honeytoken nur von einer einzigen Mitarbeiterin bzw. einem einzigen Mitarbeiter getätigt werden kann. Wird ein Teil der Datenbank gehackt und veröffentlicht, kann das Leck zur verantwortlichen Mitarbeiterin bzw. zum verantwortlichen Mitarbeiter zurückverfolgt werden. [35, S. 132]

- **Portscan enttarnen** – Durch den Einsatz von Darknets können Scans innerhalb des Netzwerkes enttarnt werden. Darknets sind IP-Ranges, in denen sich keine Server oder Computer befinden. Wird eine große Anzahl an Paketen an diese Range versendet, sollten weitere Nachforschungen angestellt werden. [46, S. 94]

- **Personen enttarnen, die das Netzwerk überwachen** – Es kann sich als schwierig bis unmöglich erweisen, Personen in einem Netzwerk zu enttarnen, die still und heimlich den vorbeikommenden Netzwerkverkehr abhören. Um diese dazu zu bringen, sich zu verraten, bieten sich folgende Möglichkeiten an: [49]
  - **Täuschung basierend auf dem Einsatz von unsicheren Protokollen** – Ein Tool könnte unverschlüsselte Logins über Protokolle wie POP3, Telnet, FTP und HTTP durchführen. Als Account-Namen kämen zum Beispiel „admin“ oder „CIO“ infrage. Möchten sich die AngreiferInnen mit einem dieser Accounts verbinden, so müssen sich diese aktiv mit dem

Netzwerk verbinden und einen Teil ihrer Tarnung aufgeben. Sobald die Anmeldung erfolgt ist, weiß das Unternehmen, dass jemand den Netzwerktraffic mitliest.

  - **Täuschung basierend auf DNS** – Es könnten vermeintlich interessante TCP-Verbindungen als Honeytokens in einem Netzwerksegment platziert werden. Wenn die AngreiferInnen nun die Hostnamen zu den IP-Adressen herausfinden möchten, führen sie einen Reverse DNS Lookup durch. Es wird Alarm geschlagen, sobald dieser Reverse Lookup stattfindet, da niemand diese Adresse abfragen darf.

- **Unautorisierter Zugriff auf WLAN enttarnen**
  - **Platzierung von Access Points mit WEP-Verschlüsselung** – Eine WEP-Verschlüsselung sollte für WLANs nicht mehr eingesetzt werden, da sie als einfach zu knacken gilt. [50] Nach wie vor gibt es aber Unternehmen, die aus Kompatibilitätsgründen Netzwerke mit WEP-Verschlüsselung betreiben müssen. Es könnten daher mehrere WEP Access Points innerhalb des Unternehmens platziert werden. Es ist davon auszugehen, dass AngreiferInnen solch ein Netzwerk knacken können. Findet plötzlich eine neue Kommunikation wie ein Portscan statt, so kann ein Alarm generiert werden.
  - **Platzierung von Access Points mit verwundbarer Konfiguration** – Es könnte ein Access Point mit einem konfigurierten Standardpasswort platziert werden. Sobald auf das Interface zugegriffen wird oder ein Netzwerkverkehr stattfindet, kann Alarm geschlagen werden.

3.7.3. Applikationsbasierte Szenarien

Die nachfolgenden Szenarien basieren auf der Annahme, dass AngreiferInnen auf eine Applikation zugreifen, die auf einem Server oder auf einer Workstation installiert ist.

- **E-Mail-Programm** – Honeytokens können auch zur Erkennung von Malware eingesetzt werden. So schlagen Cossu et al. vor, ein E-Mail-Konto einzurichten, das aber nicht verwendet wird. Werden von diesem Konto E-Mails versendet oder empfangen, kann davon ausgegangen werden, dass sich Malware auf dem System befindet. [51, S. 39]

- **Browser** – In Browsern können interessante Chroniken, Passwörter oder Lesezeichen hinterlegt werden. [44, S. 20]

- **Programme für Remote-Zugriff** – Applikationen wie SSH-, FTP- oder RDP-Clients speichern oftmals Zugangsdaten zu Remote-Systemen. [44, S. 20]

- **Ausführen von Programmen** – System- oder Anwendungsprogramme (z. B. GCC Compiler), die AngreiferInnen starten könnten, autorisierte UserInnen aber nicht starten würden. [45, S. 118]

3.7.4. Zugangsdatenbasierte Szenarien

Die nachfolgenden Szenarien beziehen sich auf die Platzierung und Verteilung von Zugangsdaten in verschiedenen Systemen und Dateien.

- **Passwortmanager** – Ein Passwortmanager könnte mit einem nicht allzu starken Passwort verschlüsselt werden. Können AngreiferInnen dieses knacken, verwenden sie die darin enthaltenen Zugangsdaten und verraten sich.
- **Windows Credential Manager** – Der Credential Manager kann nicht als sicher angesehen werden, eignet sich aber gut dafür, eine Falle zu stellen. Darin können beliebige Services mit IP-Adresse und Zugangsdaten abgelegt werden. AngreiferInnen laden dann den Inhalt des Credential Managers mit Tools wie Mimikatz herunter. Werden die hinterlegten Zugangsdaten verwendet, so kann Alarm geschlagen werden. [44, S. 22]
- **Platzierung von Zugangsdaten** – Mittels Honeytokens können Zugangsdaten zu Systemen in Dateien, Datenbanken, Webseiten und an vielen anderen Orten als Spur ausgelegt werden. Finden AngreiferInnen diese und versuchen sie, sich damit anzumelden, so kann ein Alarm generiert werden.
- **Honeywords – geknackte Passwörter sichtbar machen** – Es werden Passwort-Hashes als Köder verteilt. Haben AngreiferInnen diese geknackt und versuchen sie, sich mit dem Passwort an einem System anzumelden, so wird ein Alarm generiert. [52, S. 145]

4. Lure Box

Dieses Kapitel beschreibt die Gedanken hinter der Lure Box und erklärt ein umfassendes Szenario für deren Verwendung, um unerwünschte Zugriffe in einer Datenbank erkennen zu können.

4.1. Über die Lure Box

Die Idee hinter der Lure Box besteht darin, ein funktionstüchtiges Sicherheitsinformations- und Ereignismanagementsystem (Security Information and Event Management, SIEM) aufzusetzen, das kostenlos ist und von den jeweiligen AnwenderInnen einfach erweitert werden kann. SIEM-Systeme werden verwendet, um einen ganzheitlichen Überblick über die IT-Sicherheit in einem Unternehmen zu bekommen. Zu diesem Zweck werden sämtliche sicherheitsrelevanten Events von Servern, Netzwerk-Equipment und Arbeitsgeräten an ein zentrales SIEM-System gesendet, dort verarbeitet und abgespeichert. Die abgespeicherten Logs werden aggregiert, und mittels Regeln können Alarme generiert werden. Wurde ein Alarm ausgelöst, so können sich TechnikerInnen mit dem SIEM-System verbinden und die Logs noch einmal manuell überprüfen, ob tatsächlich eine unerwünschte Handlung durchgeführt wurde. [53] Bekannte kommerzielle SIEM-Systeme sind Splunk [54], HP ArcSight [55] oder QRadar [56].

In der vorliegenden Arbeit wurde der Fokus des SIEM-Systems auf die Erkennung von Zugriffen auf Honeytokens gelegt. Das System selbst kann aber auch für andere SIEM-Szenarien verwendet werden. Die Idee hinter der Lure Box setzt sich aus folgenden Teilen zusammen:

- **Virtuelle Maschine** – Eine fertig installierte und konfigurierte virtuelle Maschine, die ein funktionstüchtiges SIEM-System bereithält.
- **Dokumentation** – Eine Dokumentation darüber, wie die einzelnen Komponenten installiert und konfiguriert wurden.
- **Szenario** – Ein umfassendes Szenario mit der Platzierung von Honeytokens in verschiedenen Datenbanken wurde umgesetzt.

- **Mögliche Anwendungsszenarien** – Bereitstellung von Ideen zu verschiedenen Szenarien, die umgesetzt werden könnten. Mithilfe der Dokumentation und des bereits vorhandenen Szenarios sollte die Umsetzung weiterer Szenarien möglich sein.

Ein großer Vorteil des vorgeschlagenen Systems ist dessen einfache Erweiterbarkeit. Die Anbindung zahlreicher Logquellen bietet die Möglichkeit, Ereignisse über Systemgrenzen hinweg zu verfolgen. So können beispielsweise auch die Logs von anderen Servern zur Lure Box hinzugefügt werden.

4.2. Szenario

Das in der vorliegenden Arbeit verwendete Szenario ist in zwei Teile untergliedert. Der obere Ausschnitt von Abbildung 4.1 bezieht sich auf den nicht legitimen Zugriff auf Tabellen in einer Datenbank, der untere Ausschnitt auf die Verwendung von Honeytoken-E-Mail-Adressen, die in der Datenbank platziert werden.

Im oberen Szenario wird einer bestehenden Datenbank eine Salary-Tabelle hinzugefügt und mit fiktiven Gehältern von MitarbeiterInnen befüllt. Jeglicher Zugriff auf diese Tabelle wird überwacht, da es keine legitimen Gründe für einen Zugriff gibt. Sieht sich eine Person nun diese Tabelle an, so speichert die Datenbank den Zugriff mittels Logdaten ab, die an die Lure Box übertragen, dort verarbeitet und in einer Datenbank abgespeichert werden. Ein in der Lure Box laufendes Programm erkennt, dass ein nicht legitimer Zugriff erfolgt ist, und versendet mittels einer E-Mail-Nachricht einen Alarm an das Incident-Response-Team im Unternehmen. Dieses kann daraufhin weitere Untersuchungen starten.

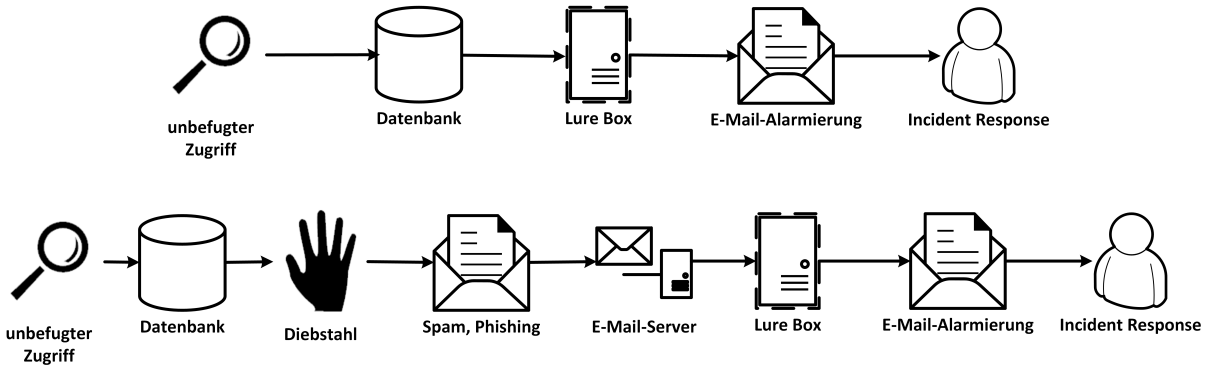

Abbildung 4.1.: Verwendung der Lure Box in zwei verschiedenen Szenarien.

Der untere Ausschnitt von Abbildung 4.1 zeigt ebenfalls einen unbefugten Zugriff auf die Datenbank. Dieses Mal wird jedoch davon ausgegangen, dass auf die Stammdaten von MitarbeiterInnen zugegriffen wird und dass diese heruntergeladen werden. In den Stammdaten befinden sich auch die E-Mail-

Adressen der MitarbeiterInnen. Zur Erkennung von Angriffen wurden den echten Daten mehrere Fake-MitarbeiterInnen hinzugefügt. Wird nun eine E-Mail-Nachricht wie zum Beispiel Spam oder Phishing an eine der Fake-E-Mail-Adressen gesendet, so wird diese vom E-Mail-Server entgegengenommen, der Informationen über alle eingehenden E-Mail-Nachrichten in Logdaten speichert. Diese Logs werden wiederum an die Lure Box weitergeleitet, dort verarbeitet und in einer Datenbank abgelegt. Ein Programm erkennt nun, dass eine E-Mail-Nachricht an eine Adresse gesendet wurde, die niemandem von außerhalb des Unternehmens bekannt sein darf. Ist die Absenderadresse unbekannt, so wird mittels einer E-Mail-Nachricht ein Alarm an das Incident-Response-Team im Unternehmen gesendet, das daraufhin weitere Untersuchungen starten kann.

4.3. Umsetzung

Abbildung 4.2 zeigt die Umsetzung des in Kapitel 4 vorgestellten Konzeptes. Wie zu sehen ist, sind die vier Datenbanksysteme Oracle-Datenbank, Microsoft SQL Server, PostgreSQL und MongoDB im Einsatz. Als E-Mail-Server wurde ein Postfix-Server installiert.

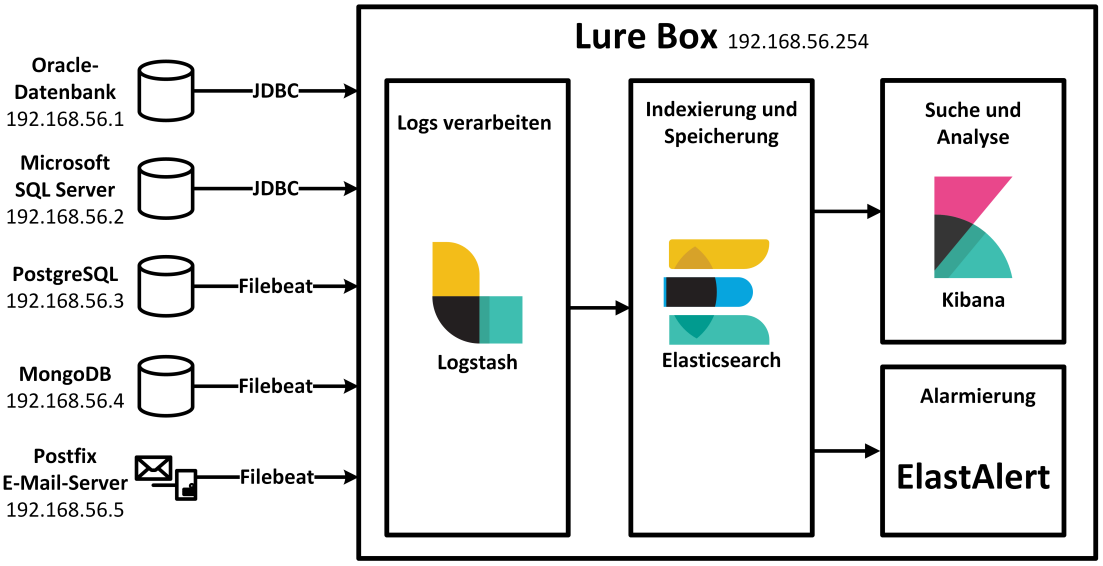

Abbildung 4.2.: Anbindung der verschiedenen Logquellen an die Lure Box. In dieser werden die Logs verarbeitet und gespeichert. Basierend auf definierten Regeln können Alarme generiert werden.

Auf allen Systemen wurde Auditing aktiviert und konfiguriert, damit die entsprechenden Logs generiert werden. Die genaue Konfiguration der einzelnen Systeme ist in den nachfolgenden Kapiteln zu finden. Über die zwei Technologien JDBC oder Filebeat [57] können die Logs zur Lure Box übertragen werden. Dort ist das Programm Logstash [58] konfiguriert, das die Logs entgegennimmt, verarbeitet und

in der Elasticsearch-Datenbank [59] langfristig speichert. Mit dem Such- und Analysetool Kibana können die empfangenen Logdaten manuell durchsucht werden. [60] Das Programm ElastAlert ist für die automatische Alarmierung zuständig. [61]

4.4. Chinook-Datenbank

Damit das Szenario und die generierten Logdaten möglichst real sind, wurde den Datenbanksystemen mit Beispieldatenbanken Leben eingehaucht. Für alle relationalen Datenbanken konnte die vom Entwickler Luis Rocha unter der MIT-Lizenz veröffentlichte Chinook-Datenbank verwendet werden. Sie steht für Microsoft SQL Server, Oracle, MySQL und viele weitere Systeme zur Verfügung. Daher ist sie für einen raschen Import in die oben genannten Systeme geeignet. Das Datenmodell repräsentiert ein Geschäft für digitale Medien und enthält daher unter anderem Tabellen für KünstlerInnen, Alben, Rechnungen und KundInnen. [62] Die Abbildung 4.3 zeigt die Abhängigkeiten der einzelnen Tabellen in der Chinook-Datenbank.

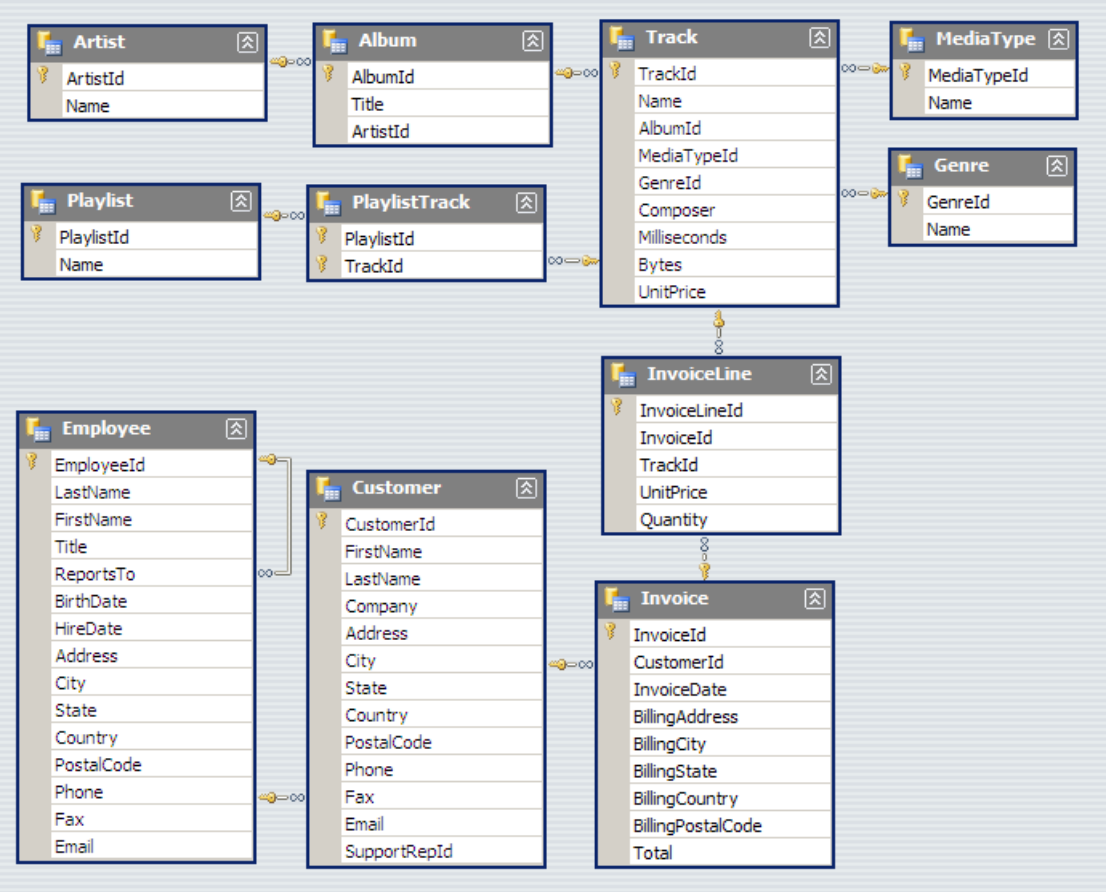

Abbildung 4.3.: Schema der Beispieldatenbank Chinook. [63]

5. Grundlagen

Dieses Kapitel vermittelt die notwendigen Grundlagen, um der in den nachfolgenden Kapiteln vorgestellten Implementierung des Szenarios folgen zu können. So werden die Unterschiede zwischen relationalen und NoSQL-Datenbanken sowie der Begriff „Auditing“ behandelt. Zusätzlich werden die drei Programme VirtualBox, Supervisor und Filebeat vorgestellt, die auf mehreren unterschiedlichen Systemen installiert und verwendet werden.

5.1. Relationale vs. NoSQL-Datenbanken

5.1.1. Relationale Datenbanken

Bereits im Jahr 1970 definierte Edgar F. Codd das relationale Datenbankmodell. Gemäß diesem sind in einer relationalen Datenbank die Daten mithilfe eines relationalen Modells organisiert. [64, S. 380] In solch einem Modell werden die Daten in Form von Tabellen (Relationen) dargestellt. Eine Relation besteht aus einer Menge von benannten Attributen, den Spalten einer Tabelle. Jedem Attribut wiederum ist ein Typ wie z. B. Integer oder String zugewiesen. Eine Zeile in einer Tabelle wird „Tupel“ genannt. Wenn eine Relation N Attribute hat, so hat jedes Tupel genau N Komponenten (eine für jedes Attribut). Jede dieser Komponenten muss vom selben Typ sein wie das dazugehörende Attribut. [65, S. 2] Der Name einer Relation zusammen mit deren Attributen wird „Schema“ genannt. [66, S. 6] Eine allgemeine Repräsentation des relationalen Datenbankmodells ist in Abbildung 5.1 zu sehen.

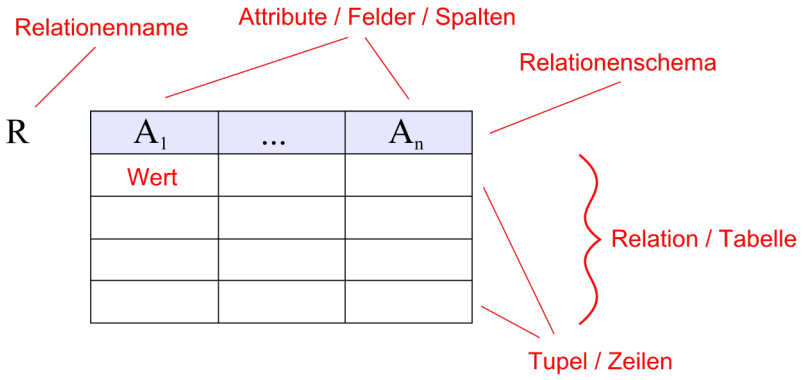

Abbildung 5.1.: Begriffe relationaler Datenbanken. [67]

Relationale Datenbanken unterstützen in der Regel die Datenbanksprache SQL. Mit dieser können unter anderem Daten eingefügt, abgefragt und gelöscht werden. Auch ist es damit möglich, Änderungen an Schemas durchzuführen. [65, S. 2]

5.1.2. NoSQL-Datenbanken

Für eine lange Zeit wurden relationale Datenbanksysteme in sehr vielen verschiedenen Anwendungsbereichen eingesetzt. Die Anforderungen in einigen dieser Bereiche haben sich im Laufe der Zeit verändert und konnten dadurch nicht mehr von den herkömmlichen Systemen erfüllt werden. [68, S. 15] Aus diesem Grund treten in den letzten Jahren vermehrt NoSQL-Datenbanken in Erscheinung. [65, S. 3] Der Begriff „NoSQL“ (Not only SQL) wurde im Jahr 2009 von Eriv Evans erfunden. NoSQL beschreibt Datenbanken mit einem Modell, bei dem das Speichern und Abrufen von Daten anders aussieht als bei relationalen Datenbankmanagementsystemen (RDBMS). NoSQL ist die Suche nach Alternativen zu relationalen Datenbanken und ist nicht gegen diese ausgerichtet. [65, S. 3]

Nachfolgend sind häufige Charakteristiken von NoSQL-Datenbanken aufgeschlüsselt: [65, S. 4–5]

- Können große Mengen von Daten besser verarbeiten als viele RDBMS.
- RDBMS wurden für vertikale Skalierung konzipiert. Damit ist gemeint, dass die Hardware des Servers, auf dem die Datenbank läuft, aufgebessert wird, wenn mehr Leistung benötigt wird. Bei NoSQL-Datenbanken wird sehr stark auf eine horizontale Skalierung gesetzt; bei dieser werden mehr Maschinen einem Cluster hinzugefügt.
- Sehr viele NoSQL-Datenbanken sind Open-Source-Projekte und können daher kostenlos verwendet werden.
- Die Struktur von NoSQL-Daten ist nicht genau mit einem Schema definiert, wie dies bei RDBMS der Fall ist.
- RDBMS unterstützen zum Großteil SQL, was bei NoSQL eine Ausnahme darstellt. Die meisten NoSQL-Systeme haben ein eigenes Interface zur Durchführung von Abfragen.

NoSQL-Datenbanken können je nach eingesetztem Datenmodell in die in Tabelle 5.1 dargestellten Arten eingeteilt werden.

Art der Datenbank	Beschreibung
Key-Value	Eines der einfachsten Datenmodelle, bei dem jeder Eintrag als Schlüssel mit seinem Wert abgespeichert wird. **Beispiele**: Amazon Dynamo, Riak und Redis.
spaltenorientiert	Sind konzeptionell den relationalen Datenbanken am ähnlichsten, da die Daten in einer Tabelle gespeichert werden. Die Datenbank speichert Inhalte spaltenweise statt zeilenweise. **Beispiele**: Google BigTable, Cassandra und HBase.
dokumentorientiert	Das Datenmodell basiert auf sogenannten „Dokumenten“. Diese sind komplexe Strukturen, die Daten als Key-Values speichern. Sie können auch viele Key-Value-Paare, Key-Array-Paare und sogar verschachtelte Dokumente beinhalten. **Beispiele**: MongoDB, Apache CouchDB und Amazon SimpleDB.
Graphen	Diese sind am besten dafür geeignet, Daten zu speichern, deren Relationen sich am besten mittels eines Graphen darstellen lassen. So wäre die Repräsentation von Netzwerktopologien oder sozialen Netzwerken mit Graphendatenbanken ideal. **Beispiele**: Neo4J und HyperGraphDB.

Tabelle 5.1.: Arten, in die NoSQL-Datenbanken eingeteilt werden können. [69, S. 5]

5.2. Auditing in Datenbanken

Damit die Lure Box die benötigten Logs bekommt, muss bei den Datenbanken das Auditing aktiviert werden. Unter „Auditing“ versteht man die Überwachung und Aufzeichnung von zuvor definierten Aktionen. Im Kontext von Datenbanken können beispielsweise SELECT-, INSERT-, UPDATE- oder DELETE-Abfragen auf bestimmte Tabellen überwacht werden. [70] Bei der Konfiguration von Auditing sollten die Best-Practice-Anleitungen der jeweiligen Hersteller beachtet werden. So besteht eine häufig verwendete Strategie darin, alle jene Informationen zu sammeln, die für die Erfüllung von Compliance-Anforderungen notwendig sind. Der Fokus sollte aber auf jene Aktivitäten gelegt werden, bei denen die meisten Sicherheitsbedenken auftauchen. Beispielsweise macht es keinen Sinn, sämtliche Tabellen einer Datenbank vollständig zu überwachen. Die Überwachung von sensiblen Tabellen, z. B. eine Gehaltstabelle, ist hingegen durchaus sinnvoll. [70]

AdministratorInnen einer Datenbank sollten stets wissen, welche BenutzerInnen sich mit einer Datenbank zu welchem Zweck verbinden dürfen. Etliche AdministratorInnen haben jedoch keine vollständige

Kontrolle über die BenutzerInnen und deren Berechtigungen in einer Datenbank. In vielen Organisationen wird ihnen vom Unternehmen vorgegeben, welche Rechte gewisse BenutzerInnenkonten haben. [71, S. 389]

Nur, wenn die Funktionsweise und Architektur eines Systems verstanden wird, kann ein sinnvolles und sicheres Auditing konfiguriert werden. Das Überwachen jeglicher Aktionen in einem System ist keine gangbare Option, da dadurch eine viel zu große Menge an Logdaten generiert werden würde. Fabri et al. schlagen für das Auditieren von Daten in einer Datenbank die Verwendung von Triggern vor, die eine Aktion durchführen, wenn ein SELECT-Statement auf bestimmte Tabellen in einer Datenbank angewendet wird. [72, S. 1141] In dieser Arbeit werden jedoch die von den Herstellern angepriesenen Auditing-Funktionen verwendet. Es wird davon ausgegangen, dass deren Sicherheit, Funktionalitäten und Performance besser sind als die eines Triggers.

5.3. VirtualBox

VirtualBox ist eine von der Firma Oracle entwickelte Virtualisierungssoftware, die unter der GNU General Public License (GPL) Version 2 zur Verfügung steht. [73] Alle an der Umsetzung beteiligten Systeme wurden als virtuelle Maschinen angelegt und mittels eines Host-only-Netzwerkes von VirtualBox miteinander verbunden, das über die GUI (Graphical User Interface) von VirtualBox konfiguriert werden kann. Dazu wählt man *File* → *Preferences* → *Network* → *Host-only network* → *(+)Add host-only network*. [74] In dem Fenster kann die in Abbildung 5.2 dargestellte IP-Adresse 192.168.56.253/24 konfiguriert werden. Sobald das Netzwerk erstellt wurde, kann es den einzelnen virtuellen Maschinen als Adapter zugewiesen werden, wodurch diese direkt miteinander kommunizieren können.

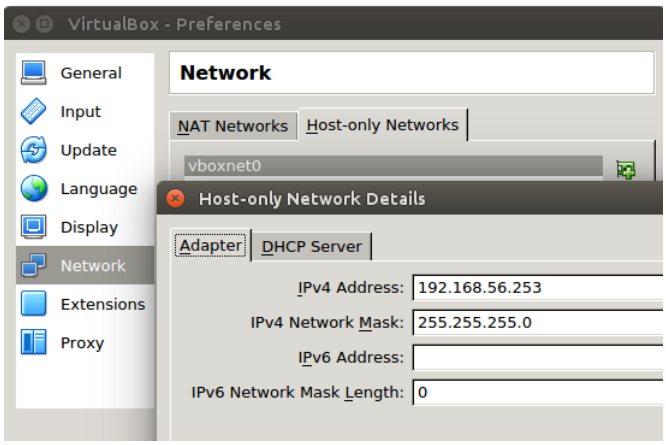

Abbildung 5.2.: Konfiguration eines Host-only-Netzwerkes mittels VirtualBox.

5.4. Supervisor

Supervisor ist ein Prozess-Kontrollsystem, das auf allen linuxbasierten Systemen installiert wurde. Es startet Prozesse als Unterprozesse und kann diese automatisch neu starten, wenn sie abstürzen. Prozesse können auch automatisch zusammen mit dem Betriebssystem gestartet werden. Die wichtigsten Gründe, warum Supervisor eingesetzt wird, sind seine Einfachheit und die Zentralisierung: Die Konfigurationsdateien von Supervisor sind sehr leicht verständlich und es werden alle relevanten Prozesse an einem zentralen Ort konfiguriert. [75]

Supervisor besteht unter anderem aus folgenden zwei wichtigen Komponenten: [75]

- **supervisord** – Ist dafür verantwortlich, die konfigurierten Prozesse zu starten, auf die Kommandos von BenutzerInnen zu reagieren, abgestürzte Prozesse neu zu starten sowie *stdout-* und *stderr-*Ouput zu loggen. Die Konfigurationsdateien der einzelnen Programme werden im Verzeichnis */etc/supervisor/conf.d/* abgelegt.
- **supervisorctl** – Der Kommandozeilen-Client von Supervisor. Mit einem shell-ähnlichen Interface kann der supervisord damit gesteuert werden. Es können der Status der laufenden Prozesse angesehen (*status*) sowie Prozesse gestartet (*start*) oder gestoppt (*stop*) werden.

Die Installation von Supervisor ist über Paketquellen möglich. Listing 5.1 zeigt die Installation über die Paketquellen und eine beispielhafte Konfigurationsdatei zum Starten von Kibana. Wie in der Konfiguration zu sehen ist, können die UserInnen angegeben werden, unter denen der Prozess gestartet werden soll. Mit *command* kann das Kommando angegeben werden, mit dem das Programm gestartet wird, und mittels *process_name* kann der Name des Prozesses definiert werden. Über den Kommandozeilen-Client kann die Konfiguration eingelesen und angewendet werden.

```
# install supervisor
# apt-get install supervisor

# example of a configuration file to start Kibana
# /etc/supervisor/conf.d/kibana.conf
[program:kibana]
user=kibana
command=/usr/share/kibana/bin/kibana  -c /etc/kibana/kibana.yml
process_name=kibana
autostart=true
autorestart=true
startsecs=15
stopsignal=INT
stopasgroup=true
killasgroup=true
stderr_logfile=/var/log/kibana/kibana.stderr.log
stdout_logfile=/var/log/kibana/kibana.stdout.log
# tell Supervisord to reread the config files
supervisorctl reread
# tell Supervisord to apply the new config files
supervisorctl update
# show the status of the processes
supervisorctl status
# start a process
supervisorctl start PROCESS_NAME
```

Listing 5.1: Beispiel für die Installation und Konfiguration von Supervisor.

Wichtig ist, dass die Ordner, in die *stdout* und *stderr* geschrieben werden sollen, existieren, ansonsten würde der Prozess nicht starten.

5.5. Filebeat

Filebeat ist ein von der Firma Elasticsearch entwickeltes Tool für die Übertragung von Logdateien. Sobald das Programm installiert ist, können Verzeichnisse, in denen sich Logdateien befinden, überwacht werden. Filebeat speichert für jede überwachte Datei die zuletzt verarbeitete Logzeile ab und kann daher immer, wenn sich eine Änderung ergibt, diese an Elasticsearch oder an Logstash weiterleiten. Wird Filebeat gestartet, so werden, wie in Abbildung 5.3 abgebildet, gleichzeitig ein oder mehrere Prospector gestartet, die innerhalb von definierten Verzeichnissen nach Logdateien suchen. Für jede gefundene Logdatei wird ein Harvester gestartet, der die Logdateien auf neuen Inhalt überprüft und Logdaten an den Spooler sendet. Beim Spooler werden die Events aggregiert und die aggregierten Daten dann beispielsweise an Logstash weitergeleitet. [57]

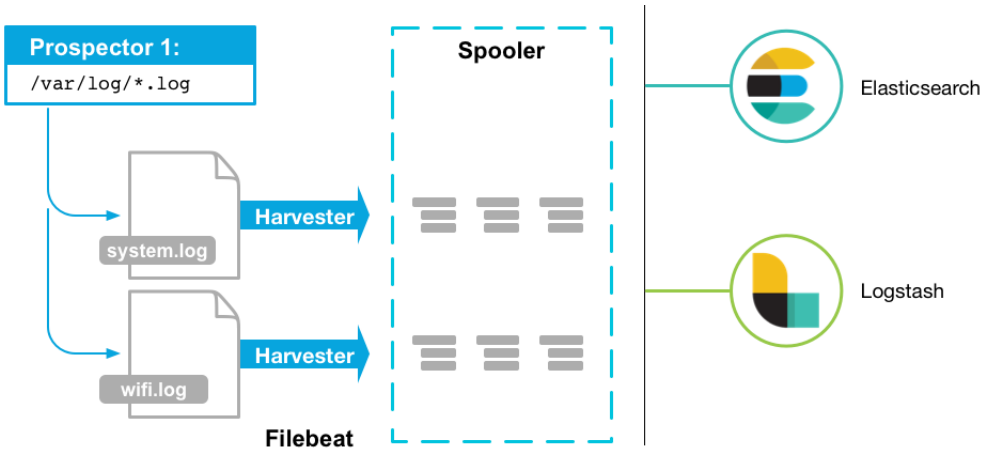

Abbildung 5.3.: Die Funktionsweise von Filebeat. [57]

Filebeat kann, wie in Listing A.2 dargestellt, über die Paketquellen installiert werden. [76] Ist Filebeat installiert und konfiguriert, so werden Informationen über die letzte Position in den Logdaten unter */var/-lib/filebeat/registry* abgespeichert. Für die Konfiguration selbst ist die YML-Konfigurationsdatei */etc/filebeat/filebeat.yml* wichtig.

6. Lure Box – Installation und Konfiguration

Dieses Kapitel beschäftigt sich mit der Installation und Konfiguration der Software für die Lure Box. Auf ihr wurden neben dem Debian-Betriebssystem die Programme Elasticsearch, Cerebro, Logstash, Kibana, Supervisor und ElastAlert installiert.

6.1. Elasticsearch

Elasticsearch ist eine von der Firma Elasticsearch entwickelte, hoch skalierbare Datenbank, mit der große Mengen an Daten beinahe in Echtzeit gespeichert, gesucht und analysiert werden können. Im Szenario wird Elasticsearch dazu verwendet, die verarbeiteten Logdaten abzuspeichern. [59]

Für den Umgang mit Elasticsearch hilft es, die folgenden Konzepte zu verstehen: [77]

- **Cluster** – Ein Cluster ist eine Sammlung von einem oder mehreren Nodes (Servern), die zusammen alle Daten speichern. Ein Cluster wird über einen eindeutigen Namen identifiziert, der standardmäßig *Elasticsearch* lautet.
- **Node** – Ein Node ist ein einzelner Server, der Teil des Clusters ist. Wie ein Cluster verfügt auch ein Node mit *Universally Unique IDentifier (UUID)* über einen eindeutigen Namen. In einem Cluster können beliebig viele Nodes vorhanden sein.
- **Index** – Ein Index ist eine Sammlung von Dokumenten, die eine Gemeinsamkeit besitzen. So kann es beispielsweise einen Index für Oracle-Logs geben und einen für MongoDB-Logs. Ein Index wird über einen Namen identifiziert, der für das Speichern, Updaten und Löschen von Daten benötigt wird.
- **Dokument** – Ein Dokument ist jenes Element, das in einem Index gespeichert werden kann. Jede Logzeile wird zu einem Dokument in Elasticsearch. Das Dokument selbst wird im JSON-Format abgespeichert.

Für die Installation von Elasticsearch muss mindestens Java 8 auf dem System vorhanden sein. Nach der Java-Installation kann Elasticsearch aus den Paketquellen von Elasticsearch installiert werden. Diese müssen zuvor, wie in Listing A.1 beschrieben, hinzugefügt werden. Listing A.3 zeigt die Installation von Elasticsearch und die dafür benötigten Abhängigkeiten. An der Konfigurationsdatei von Elasticsearch */etc/elasticsearch/elasticsearch.yml* wurden keine Änderungen vorgenommen.

6.2. Cerebro

Cerebro ist ein unter der MIT-Lizenz veröffentlichtes Open-Source-Tool zur Administration von Elasticsearch über ein Webinterface. Es basiert auf Scala, dem Play Framework, AngularJS und Bootstrap. Für die Ausführung von Cerebro ist Java 1.8 oder eine neuere Version erforderlich. [78]

Listing A.4 zeigt die notwendigen Schritte für die Installation von Cerebro. Ist diese abgeschlossen, so wird die Konfigurationsdatei */opt/cerebro/cerebro-0.6.4/conf/application.conf* angepasst. Darin wird in der *hosts*-Sektion die lokale Elasticsearch-Instanz hinzugefügt, wodurch diese beim Aufrufen der Cerebro-Webseite direkt ausgewählt werden kann. Nach der Konfiguration kann Cerebro mittels Supervisor gestartet werden. Listing A.11 zeigt die dafür notwendige Konfigurationsdatei */etc/supervisor/conf.d/cerebro.conf*. Nach dem Start ist über die URL `http://192.168.56.254:9000/` der Zugriff auf Cerebro möglich.

6.3. Logstash

Logstash ist ein von der Firma Elasticsearch entwickeltes Programm zum Sammeln, Verarbeiten und Abspeichern von Logs. [79] Die Installation ist unter Debian problemlos über die Paketquellen möglich. [80] Diese müssen zuvor, wie in Listing A.1 beschrieben, hinzugefügt werden. Die Installation von Logstash wird in Listing A.5 dargestellt. Darin wird ein Ordner namens *last_run* angelegt, der von verschiedenen Plug-ins dazu verwendet wird, Informationen über die zuletzt abgeholten Events zu speichern.

Abbildung 6.1 zeigt die drei Teile der Logstash Pipeline. Im Input werden die Events in Logstash hineingeladen, im Filter werden sie modifiziert und mit dem Output können sie an eine bestimmte Destination, wie zum Beispiel Elasticsearch, gesendet werden. [81] Jedes einzelne Event läuft – unabhängig von bereits verarbeiteten Events – durch diese Pipeline.

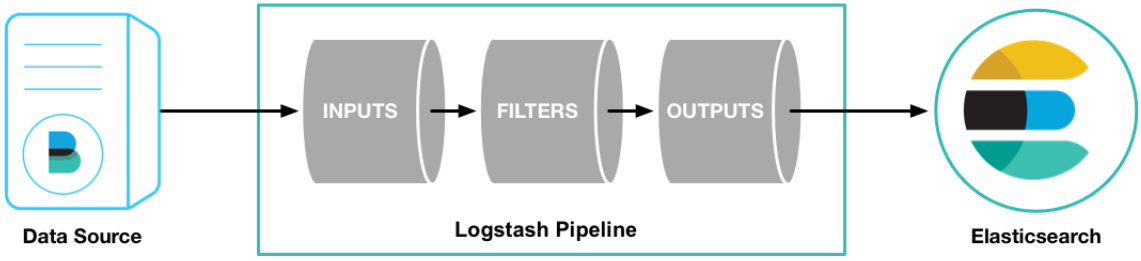

Abbildung 6.1.: Die Logstash Verarbeitungspipeline. [79]

6.3.1. Input

Um die Logs zur Lure Box zu übertragen, bieten sich verschiedene Möglichkeiten an. [82] Logstash unterstützt eine Vielzahl von sogenannten „Input Plug-ins“, mit denen Logdaten von verschiedenen Quellen gelesen werden können. [82] Nachstehend findet sich ein Auszug von verfügbaren Input Plug-ins: [82]

- **beats** – Zur Annahme von Logs, die über Filebeat gesendet werden.
- **file** – Events können aus einer Logdatei gestreamt werden.
- **jdbc** – Daten können von Datenbanken über das JDBC-Interface abgefragt werden.
- **stdin** – Liest Events von der Standardeingabe.
- **tcp** – Liest Events von einem TCP-Socket.
- **udp** – Liest Events von einem UDP-Socket.

6.3.2. Filter und Output

Filter erlauben es, die empfangenen Logs zu verarbeiten. Mithilfe von bedingten Anweisungen und Verzweigungen können Events geändert, gelöscht oder mit weiteren Informationen angereichert werden. [81] Nachfolgend findet sich ein Auszug an verfügbaren Filtern, die im Zuge der Implementierung verwendet wurden: [83]

- **csv** – Der CSV-Filter nimmt ein Event-Feld, das CSV-Daten enthält, und teilt dieses in die darin enthaltenen individuellen Felder auf. [84]
- **date** – Bevor Logstash ein Event in den Output speichert, wird ein Feld namens *@timestamp* hinzugefügt. Dieses ist von zentraler Bedeutung, da Events innerhalb von Elasticsearch anhand dieses Feldes sortiert werden. Mit dem Date-Filter kann ein Zeitstempel aus dem Logevent verwendet werden, um das Timestamp-Feld zu definieren. Wird der Date-Filter nicht angewendet, so wählt

Logstash jenen Zeitpunkt als Zeitstempel, an dem das Event im Input-Teil gelesen wurde. [85] Ziel sollte es sein, den Zeitpunkt, an dem das Event generiert wurde, im UTC-Format in Elasticsearch zu speichern.

- **drop** – Mit diesem Filter kann ein komplettes Event verworfen werden, ohne es weiter in den Output zu leiten. [86]
- **grok** – Grok erlaubt das Parsen und Strukturieren von beliebigen Texten. Grok setzt auf eine Kombination von Patterns, die regulären Ausdrücken ähneln und passend zu den Logs gebaut werden. Über das Pattern kann definiert werden, dass ein bestimmter Teil einer Nachricht in einem anderen Feld mit einem bestimmten Datentyp abgespeichert werden soll. Ein großer Vorteil von Logstash ist, dass es bereits von sich aus mehr als 120 verschiedene Patterns für das Verarbeiten von Logs mitliefert. [87]
- **json** – Genau wie der CSV-Filter teilt dieser ein Event-Feld, das JSON-Daten enthält, in die darin enthaltenen individuellen Felder auf. [88]
- **mutate** – Mit diesem Filter können generelle Veränderungen wie Umbenennen, Entfernen und Ersetzen durchgeführt werden. [89]

Nachdem ein Event über den Input empfangen und über den Filter verarbeitet wurde, muss es an seinen Bestimmungsort geschrieben werden, wofür der Output-Teil verwendet wird. Ein Event kann bei Bedarf an mehrere Outputs, wie zum Beispiel an Elasticsearch oder an eine Datei, gesendet werden. [81]

Listing 6.1 zeigt eine Logstash-Konfiguration, bei der von der Standardeingabe (*stdin*) gelesen wird. Ein Grok-Filter parst den Inhalt des Message-Feldes. Über den Date-Filter wird der Zeitstempel aus dem Timestamp-Feld übernommen. Ist der Filter abgearbeitet, so wird der Output einmal an Elasticsearch und einmal an die Standardausgabe (*stdout*) gesendet. [90]

```
input { stdin { } }

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Listing 6.1: Beispiel der Logstash-Konfigurationsdatei *test.conf*. [90]

Mit dem in Listing 6.2 vorhandenen Befehl könnte die zuvor definierte Logstash-Konfiguration gestartet werden.

```
/usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/test.conf
```

Listing 6.2: Logstash-Konfiguration über Kommandozeile starten. [91]

Kommt es bei der Verwendung von Logstash zu Problemen, so sollte man sicherstellen, dass keine Firewall den Netzwerkverkehr blockiert, und überprüfen, ob sich in den Logstash-Logs Hinweise auf mögliche Fehlerquellen befinden.

6.4. Kibana

Kibana ist eine Open-Source-Analyse- und Visualisierungsplattform, die für die Zusammenarbeit mit Elasticsearch entwickelt wurde. Kibana kann für das Suchen und Anzeigen von in Elasticsearch gespeicherten Daten verwendet werden. Außerdem bietet sich die Möglichkeit, Daten mittels Tabellen, Karten und verschiedenen anderen Diagrammen zu visualisieren. [60]

Die Installation ist unter Debian problemlos über die Paketquellen möglich, [92] die zuvor, wie in Listing A.1 beschrieben, hinzugefügt werden müssen. Die Installation von Kibana wird in Listing A.6 dargestellt. Nach der Installation muss die Konfigurationsdatei von Kibana unter */etc/kibana/kibana.yml*

angepasst werden. Es wurde lediglich der Servername von Kibana verändert und das Programm so konfiguriert, dass es auf allen Netzwerkinterfaces auf Anfragen hört.

Da für das Verbinden mit Kibana standardmäßig keine Authentifizierung notwendig ist, kann dies ein Sicherheitsrisiko darstellen. Aus diesem Grund wird empfohlen, das kostenpflichtige Programm Elastic X-Pack [93] oder einen kostenlosen Reverse Proxy mit Authentifizierung zu verwenden. Nach der Konfiguration kann Kibana mittels Supervisor gestartet werden; Listing A.12 zeigt die dafür notwendige Konfigurationsdatei */etc/supervisor/conf.d/kibana.conf*. Nach dem Start ist über die URL `http://192.168.56.254:5601/` der Zugriff auf Kibana möglich.

6.5. ElastAlert

Damit das Incident-Response-Team eines Unternehmens über die Zugriffe oder die Verwendung von Honeytokens informiert wird, muss eine Alarmierung stattfinden. Die Firma Elasticsearch bietet dafür das kostenpflichtige Produkt „Watcher“ an. [94] Als kostenlose Alternative wurde ElastAlert ausgewählt, das von der Firma Yelp Inc. unter der Apache-Lizenz Version 2.0 bereitgestellt wird. [61]

ElastAlert ist ein Framework, mit dessen Hilfe Anomalien und gewisse Muster in den Daten erkannt werden können. Dabei setzt das Programm auf folgende drei Komponenten: [61]

- **Elasticsearch** – Wird von ElastAlert periodisch durchsucht.
- **Regeln** – Legen fest, wonach gesucht werden soll.
- **Alarme** – Legen fest, auf welche Art und Weise alarmiert werden soll, wenn eine Regel anschlägt.

Als Alarmmöglichkeiten werden unter anderem eine E-Mail, JIRA oder Telegram unterstützt. [61]

ElastAlert basiert auf Python 2.7 und kann nach dem Herunterladen aus einem GIT Repository installiert werden. [95] Damit es zu keinen Versionskonflikten zwischen den bereits installierten Python-Bibliotheken und den von ElastAlert benötigten kommt, wird eine *virtualenv*-Umgebung für ElastAlert erzeugt. Die notwendigen Schritte zur Installation von ElastAlert werden in Listing A.7 dargestellt; darin wird auch ein Index für ElastAlert in Elasticsearch angelegt, dessen Funktion darin besteht, Informationen und Metadaten über Abfragen und Alarme in ElasticSearch zu speichern. [95]

Die Konfigurationsdatei */etc/elastalert/config.yaml* musste an die in Listing A.8 dargestellten Werte angepasst werden. Wie zu sehen ist, sucht ElastAlert von nun an in dem Rules-Ordner nach Regeln. Elasticsearch wird jede Minute von ElastAlert abgefragt. Die Option *buffer_time* teilt ElastAlert mit, wie weit vom aktuellen Zeitpunkt aus zurückliegende Events in der Vergangenheit überprüft werden sollen. Für die Festlegung des Zeitraums kommt das Timestamp-Feld in Elasticsearch zum Einsatz. [95] Am Ende der Konfigurationsdatei ist eine Sektion für E-Mail-Alarme zu sehen, in welcher der SMTP-Server zum Versenden von E-Mails definiert wird. Zum Abspeichern der Zugangsdaten wird eine *.yaml*-Datei verwendet. Diese sollte den in Listing A.9 gezeigten Inhalt haben. Damit die Datei vor Zugriffen von anderen UserInnen auf dem System geschützt ist, wird eine separate Userin bzw. ein separater User angelegt, dem die erforderlichen Berechtigungen zugewiesen werden.

Mit den in Listing A.10 dargestellten Befehlen kann eine einzelne Regel getestet oder ElastAlert für sämtliche Regeln gestartet werden. [95] ElastAlert lässt sich auch mittels Supervisor starten und überwachen. Der Inhalt der dafür notwendigen Konfigurationsdatei */etc/supervisor/conf.d/elastalert.conf* findet sich unter Listing A.13.

7. Oracle-Datenbank

Die Oracle-Datenbank ist ein relationales Datenbankmanagementsystem (RDBMS) und wird seit den 1970er-Jahren von der Firma Oracle Corporation kommerziell entwickelt. [96] Gemäß dem Ranking von *db-engines.com* erreichte die Oracle-Datenbank im Mai 2017 den ersten Rang unter den populärsten verfügbaren Datenbanksystemen. [97]

7.1. Installation

Für die Implementierung wurde eine von Oracle zur Verfügung gestellte virtuelle Maschine verwendet. [98] Diese kann mittels der Virtualisierungssoftware VirtualBox [73] importiert und gestartet werden. Der Einsatz eines vorgefertigten Images ist für den für diese Arbeit erforderlichen Zweck ideal, da ein aufwändiger Installationsprozess entfällt. An dieser Stelle muss betont werden, dass die Images keinesfalls in einer produktiven Umgebung eingesetzt werden sollten. Die VM enthielt zum Zeitpunkt des Herunterladens unter anderem folgende Komponenten: [98]

- **Betriebssystem** – Oracle Linux 7
- **Datenbanksystem** – Oracle Database 12c Release 2 Enterprise Edition
- **Zusätzliche Software** – Oracle SQL Developer 4.1.5.21 (Entwicklungsumgebung für Oracle-Datenbanken)

Wird die virtuelle Maschine gestartet, so werden in einer Shell wichtige Informationen zu der bereits installierten Datenbank angezeigt. Eine Zusammenfassung dieser Werte ist in Tabelle 7.1 zu sehen.

Schlüssel	Wert
SID (Oracle System ID)	orcl12c
Service name	orcl
Hostname	localhost
Port	1521
Benutzername	system
Passwort	oracle

Tabelle 7.1.: Zugriffsdaten auf die Oracle-Datenbank.

7.2. Importieren der Chinook-Datenbank

Für das Importieren der Datenbank wurde der in Listing A.14 angeführte Befehl ausgeführt. Mittels *sqlplus* wird das Skript *Chinook_Oracle.sql* ausgeführt. Damit dieses ausgeführt werden kann, wird eine Verbindung zum Datenbanksystem mittels *sys*-User, dem Passwort *oracle* und der Rolle *sysdba* hergestellt. Die daraus resultierende Ausgabe wird in die Output-Datei umgeleitet, sodass diese später auf Fehler überprüft werden kann.

Durch den Import wird ein User namens *chinook* mit dem Passwort *p4ssw0rd* angelegt, dem verschiedene Rechte zum Anlegen von Tabellen und anderen Objekten zugewiesen werden. Anschließend verbindet sich das Skript als User *chinook* mit der Datenbank, erzeugt dort verschiedene Tabellen und fügt Zeilen in die Tabellen ein. Sobald der Importvorgang abgeschlossen ist, kann eine Verbindung mittels des neuen Users *chinook* und dem Passwort *p4ssw0rd* hergestellt werden. Zusätzlich muss noch die Salary-Tabelle hinzugefügt werden, auf die keine Person zugreifen darf. Dies wird in Listing A.15 durchgeführt; außerdem werden ein paar Einträge hinzugefügt.

7.3. Auditing in Oracle 12c

Oracle versteht unter „Auditing“ das Überwachen und Aufzeichnen von Aktionen in einer Datenbank. [70] Mittels Oracle 12c wurde die Funktionsweise des Auditings komplett neu gestaltet. In vorherigen Versionen wurden mehrere Audit-Einträge an unterschiedlichen Stellen angelegt. [99, S. 4], [100] Mit der neuen Version wurde das sogenannte „Unified Auditing“ hinzugefügt. Dabei wird jeder Audit-Eintrag an einer zentralen Stelle in einem einheitlichen Format gespeichert und kann mit der *Unified_Audit_Trail* View begutachtet werden. [70] Das Unified Auditing ist standardmäßig in der Oracle-Enterprise-Edition

enthalten. Es werden dafür keine zusätzlichen Lizenzen benötigt. [99, S. 4]

Damit den AnwenderInnen der Umstieg auf das neue Verfahren erleichtert wird, bietet Oracle die Möglichkeit, das Unified Auditing in zwei unterschiedlichen Modi zu betreiben: [70], [99, S. 4]

- **Mixed Mode** – Standardmäßig in Version 12c aktiv. Alle älteren Log- und Auditfunktionen sowie Konfigurationen arbeiten wie zuvor. Das Unified Auditing wird zusätzlich aktiviert. Dies erlaubt einen schrittweisen Übergang in den PURE-Modus. Daten können im Klartext abgespeichert werden, wenn *syslog* verwendet wird. Unified Auditing im Mixed-Modus lässt sich starten, indem mindestens eine Unified Audit Policy aktiviert wird.
- **FULL Mode, PURE Mode** – Sobald dieser Modus aktiviert ist, werden sämtliche Log- und Auditfunktionen aus älteren Versionen ignoriert. Logdaten werden in der Datenbank selbst und zusätzlich in sogenannten „Oracle SecureFiles“ auf dem Dateisystem abgespeichert. Aus diesem Grund wird der Einsatz von *syslog* nicht mehr unterstützt. In der Datenbank kann die View *SYS.UNIFIED_AUDIT_TRAIL* dazu verwendet werden, um Einsicht in die Logs zu nehmen.

Mit den sogenannten „System Global Area (SGA) Queues“ wurde ein neues Schema eingeführt, das Performanceverbesserungen bringen soll. Oracle 12c schreibt Audit-Daten zuerst in diese Queues; diese werden in periodischen Abständen geleert und in die AUDSYS-Schema-Tabelle geschrieben. Die SGA kann im Immediate-write-Modus und im Queued-write-Modus betrieben werden. Bei Ersterem werden neue Einträge sofort geschrieben, bei Letzterem werden neue Audit-Einträge periodisch weggeschrieben. Standardmäßig ist der Queued-write-Modus aktiviert. [99, S. 12], [70]

Abbildung 7.1 gibt einen Überblick, wie die *Unified_Audit_Trail* funktioniert. Wie zu sehen ist, werden gewisse Aktionen geloggt, die zuvor festgelegt werden müssen. Diese Logs landen dann in der SGA und werden von dort in periodischen Abständen in die Read-only-AUDSYS-Tabelle geschrieben. Diese Tabelle kann mittels der View *SYS.UNIFIED_AUDIT_TRAIL* angesehen werden. [101]

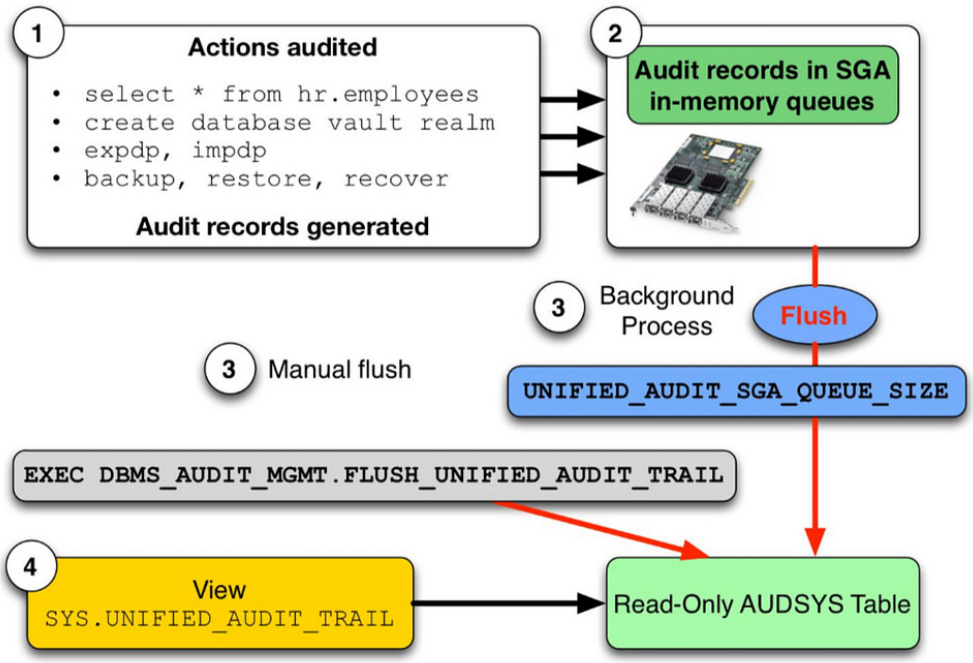

Abbildung 7.1.: Überblick über die Funktionsweise der *Unified_Audit_Trail*. [100]

Ebenfalls mit dem Unified Auditing hinzugekommen sind folgende beiden Rollen: [99, S. 11]

- **AUDIT_ADMIN** – Diese Rolle wird nur an vertrauenswürdige UserInnen vergeben und erlaubt das Erstellen und Verwalten von Audit Policies. Darüber hinaus erlaubt sie den Einsatz des AUDIT und NOAUDIT SQL Statements sowie das Ansehen der Audit-Daten.
- **AUDIT_VIEWER** – Diese Rolle wird üblicherweise an externe oder interne AuditorInnen vergeben und erlaubt lediglich das Ansehen und Analysieren von Audit-Daten.

Das Unified Auditing selbst kann über eine der folgenden Methoden konfiguriert werden: [102]

- **Erstellung einer neuen Unified Audit Policy** – Mit einer Audit Policy können Datenbankaktionen, Datenbankobjekte sowie Datenbankprivilegien überwacht werden.
- **Eine bestehende Unified Audit Policy verwenden** – Oracle liefert Standard Unified Audit Policies, die häufig benötigte Auditszenarien abdecken. Die Audit Policy *ORA_SECURECONFIG* ist ein Beispiel dafür.
- **Einsatz von Fine Grained Audit Policies** – Das Fine Grained Auditing erlaubt das Definieren von Bedingungen, die zutreffen müssen, bevor auditiert wird. Es wird ermöglicht, um Zugriffe auf bestimmte Spalten und Felder zu überwachen. Auch können damit Aktionen wie die folgenden überwacht werden:

  - Zugriff auf eine Tabelle zwischen 21:00 und 06:00 Uhr oder Zugriff an einem Wochenende
  - Zugriff mittels einer IP-Adresse von außerhalb des Unternehmensnetzwerkes
  - Selektieren oder Updaten einer Spalte in einer Tabelle
  - Verändern eines Wertes in einer Spalte in einer Tabelle

7.4. Umsetzung der Überwachung

7.4.1. Related Work

Bereits im Jahr 2004 äußerten Čenys et al. den Vorschlag, Tabellen in einer Oracle-Datenbank mittels Fine Grained Auditing zu überwachen. [103, S. 5] Zwei Jahre später schlugen sie drei unterschiedliche Methoden vor, um Honeytokens in Oracle-Datenbanken zu überwachen: [48, S. 452–453]

- **Pipeline-Funktion mit externer Prozedur** – Eine Pipeline-Funktion ist ein spezielles Objekt, das dazu verwendet werden kann, eine Datenbanktabelle zu emulieren. Wird auf die emulierte Tabelle zugegriffen, so wird eine externe Prozedur aufgerufen, die einen Alarm generiert.
- **Kombination eines Triggers mit einer externen Prozedur** – Der Trigger überwacht Zugriffe auf eine Tabelle. Wird auf diese zugegriffen, so wird eine externe Prozedur aufgerufen, die einen Alarm generiert.
- **Kombination von Oracle FGA und einer externen Library** – Mittels der externen Library wird ein Alarm generiert.

Ray et al. beschäftigten sich mit der Erkennung von Advanced Persistent Threats sowie mit Oracle-Datenbanken. Dabei wurden ausführlich jene Elemente in der Datenbank genannt, die überwacht werden müssen, um Angriffe erkennen zu können. [104]

7.4.2. Auditing-Konfiguration

Basierend auf den zur Verfügung stehenden Technologien wurde für die Oracle-Datenbank folgende Lösung erarbeitet:

- **Überwachung der Salary Tabelle** – Jeglicher Zugriff auf diese Tabelle wird mittels der Unified Audit Policy *MONITOR_SALARY* überwacht.
- **Überwachung von Honeytokens in der Employee-Tabelle** – Es wird angenommen, dass es sich bei den Employees mit der ID 4 und 5 um Honeytoken-UserInnen handelt. Jeglicher Zugriff auf diese wird mittels der Fine Grained Audit Policy *MONITOR_EMPLOYEE* überwacht.

Bevor die entsprechenden Policies erstellt werden können, muss das Unified Auditing in den PURE-Modus umgeschaltet werden. [105] Dazu müssen die in Listing A.16 beschriebenen Schritte durchgeführt werden. Anschließend kann mit der SQL-Abfrage aus Listing A.17 überprüft werden, ob die Umstellung in den PURE-Modus erfolgreich war. Kommt als Ergebnis *true* heraus, so war die Umstellung erfolgreich. Auch wird zum Testen von Auditregeln empfohlen, die SGA, wie in Listing A.18 dargestellt, in den Immediate-write-Modus umzuschalten. Dadurch kann man sofort erkennen, ob das Auditing wie gewünscht funktioniert. Sollten die Datenbanken zahlreiche Anfragen verarbeiten müssen, so sollte die SGA wieder in den Queued-write-Modus umgeschaltet werden. [106]

Listing A.19 zeigt, wie die Unified Audit Policy *MONITOR_SALARY* zur Überwachung von Zugriffen auf die Salary-Tabelle erstellt und angezeigt werden kann. Listing A.20 zeigt, wie eine Fine Grained Audit Policy zur Überwachung von Zugriffen auf die Honeytokens in der Employee-Tabelle erstellt und angezeigt werden kann.

7.4.3. Remote-Zugriff auf die Oracle-Datenbank

Damit das Szenario möglichst realistisch ist, wird ein Remote-Zugriff von der Lure-Box auf die Daten in der Oracle-Datenbank durchgeführt. Dazu bietet sich der von Oracle entwickelte Client *sqlplus* an. Dieser kann von der Oracle-Webseite heruntergeladen und anschließend, wie in Listing A.21 zu sehen, installiert werden. [107] Danach kann mit dem User *chinook* eine Verbindung zur Datenbank hergestellt werden und es können verschiedene Abfragen, wie in Listing A.22 zu sehen, durchgeführt werden.

7.4.4. Logs ansehen

Mit der *Unified_Audit_Trail* View kann man die entsprechenden Audit-Logs nach den zuvor durchgeführten Zugriffen ansehen. Greifen nun UserInnen auf die Salary-Tabelle zu, so können die generierten Logs mit den in Listing A.23 dargestellten Befehlen angesehen werden. Die Zugriffe auf die Honeytokens in der Employee-Tabelle können mit den in Listing A.24 dargestellten Befehlen angesehen werden.

7.4.5. Löschen und Archivieren von Audit Trail Records

Damit die *Unified_Audit_Trail* niemals zu voll wird, sollte die Tabelle in periodischen Abständen archiviert und gelöscht werden. Dazu kann beispielsweise ein Job angelegt werden, der die Einträge in regelmäßigen Abständen löscht. Oracle bietet die Möglichkeit, alle Audit-Trail-Einträge vor einem gewissen Datum oder sogar sämtliche Einträge zu löschen. Damit das Löschen durchgeführt werden kann,

müssen die UserInnen die Rolle AUDIT_ADMIN besitzen. Die Oracle-Datenbank zeichnet auf, wenn Einträge aus der *Unified_Audit_Trail* gelöscht wurden. [108]

7.5. Anbindung an die Lure Box

Die Lure Box wird mittels JDBC an die Oracle-Datenbank angebunden, wodurch die Logs direkt abgeholt werden können. Um dies zu ermöglichen, muss ein Konto mit der Rolle *AUIDT_VIEWER* angelegt werden. Die dafür erforderlichen Befehle sowie die Überprüfung der Verbindung von der Lure Box aus werden in Listing A.25 gezeigt.

Damit Logstash mittels JDBC auf die Datenbank zugreifen kann, wird ein entsprechender Treiber benötigt, der von der Oracle-Webseite heruntergeladen werden kann. [109] Listing A.26 zeigt, wie dieser Treiber im Verzeichnis */opt/jdbc/oracle* abgelegt wird.

Auf der Lure Box muss die Konfigurationsdatei */etc/logstash/conf.d/oracle_database.conf* angelegt werden. Ihr Inhalt ist in Listing A.27 ersichtlich. Essenziell ist die Option *record_last_run*; dadurch wird sichergestellt, dass Logstash immer nur jene Logs abholt, die noch nicht übertragen wurden. Zusätzlich muss ein Datenverzeichnis, wie in Listing A.28 zu sehen, für die neue Logstash-Instanz angelegt werden.

Nach dem Anlegen der Konfigurationsdatei kann Logstash mittels Supervisor gestartet und überwacht werden. Der Inhalt der Konfigurationsdatei */etc/supervisor/conf.d/logstash-oracle.conf*, die dafür notwendig ist, befindet sich unter Listing A.29. Ist Logstash gestartet, so werden die Logs abgeholt, verarbeitet und in Elasticsearch gespeichert. Damit auch eine E-Mail-Alarmierung erfolgt, muss eine Regel für ElastAlert erstellt werden. Listing A.30 zeigt den Inhalt der Datei */etc/elastalert/rules/monitor_oracle.yaml*.

8. Microsoft SQL Server

Im Jahr 1989 veröffentlichte das Unternehmen Microsoft zusammen mit Sybase und Ashton-Tate die SQL Server Version 1.0. [110, S. 3] Der SQL Server ist ein relationales Datenbankmanagementsystem (RDMBS), das kommerziell von Microsoft vertrieben wird. [111, S. 1] Gemäß dem Ranking von *db-engines.com* erreichte der Microsoft SQL Server im Mai 2017 den dritten Rang unter den populärsten verfügbaren Datenbanksystemen. [97]

8.1. Installation

Für die Implementierung wurde mittels der Virtualisierungssoftware VirtualBox [73] eine virtuelle Maschine mit folgenden Komponenten installiert:

- **Betriebssystem** – Windows Server 2016 Datacenter
- **Datenbanksystem** – Microsoft SQL Server 2016 Enterprise mit Service Pack 1
- **Zusätzliche Software** – SQL Server Management Studio 16.5.3 (Entwicklungsumgebung für Microsoft SQL Datenbanken) [112]

8.2. Import der Chinook-Datenbank

Für das Importieren der Datenbank wurde der in Listing A.31 angeführte Befehl ausgeführt. Mittels *sqlcmd* wird das Skript *Chinook_SqlServer_AutoIncrementPKs.sql* ausgeführt. Die daraus resultierende Ausgabe soll in die Datei *output.txt* geschrieben werden. [113]

Durch den Import wird eine Datenbank namens „Chinook“ angelegt. In dieser sind alle Tabellen und Daten vorhanden. Zusätzlich muss noch die Salary-Tabelle hinzugefügt werden, auf die keine Person zugreifen darf. Dies wird in Listing A.32 durchgeführt. Darüber hinaus werden ein paar Einträge hinzugefügt.

8.3. Auditing im Microsoft SQL Server 2016 Enterprise

Der SQL Server ermöglicht das Monitoren des Servers mittels Server-Level-Audits und Datenbank-Level-Audits. Folgende Aufzählung gibt eine Hilfestellung, um entscheiden zu können, welcher Audit-Typ für einen konkreten Fall benötigt wird: [71, S. 389]

- **Server-Level-Audit sollte gewählt werden, wenn Folgendes überwacht werden soll:**
  - Aktionen, die den gesamten Server betreffen
  - Aktionen, die Veränderungen über alle Datenbanken hinweg überwachen
  - Aktionen, die Veränderungen an den Schemas aller Datenbanken überwachen
- **Datenbank-Level-Audit sollte gewählt werden, wenn Folgendes überwacht werden soll:**
  - Aktionen, die eine Datenbank, ein Objekt oder ein Schema betreffen
  - Aktionen von UserInnen innerhalb einer Datenbank
  - bestimmte Aktionen wie SELECT, DELETE, UPDATE oder andere Data Manipulation Language (DML) Statements innerhalb einer Datenbank

Seit der Version SQL Server 2016 Service Pack 1 unterstützen alle Editionen sowohl Server-Level-Audits als auch Datenbank-Level-Audits. Zuvor war das Datenbank-Level-Auditing den Enterprise-, Developer- und Evaluation-Editionen vorbehalten. [114]

Bevor ein Server-Level-Audit oder ein Datenbank-Level-Audit festgelegt wird, muss ein sogenanntes „Server-Audit-Objekt“ erstellt werden; dieses bietet die Tools und Prozesse, die benötigt werden, um Audits zu aktivieren, zu speichern und anzusehen. [115] Bei der Definition von Server-Level-Audits oder Datenbank-Level-Audits wird daraufhin festgelegt, in welches Server-Audit-Objekt die auftretenden Logs geschrieben werden sollen. Es ist möglich, mehrere Server-Audit-Objekte per SQL Serverinstanz zu definieren. [115] Das Objekt enthält keine Informationen über die Aktionen, die überwacht werden sollen. [116] Wichtig ist, zu wissen, dass das Server-Audit-Objekt immer im deaktivierten Zustand erstellt wird und dass daher standardmäßig keine einzige Aktion geloggt wird. [115]

Weitere Informationen über Server-Level-Audit und Datenbank-Level-Audit:

- **Server-Audit-Spezifikation** – Diese besteht aus Gruppen von Server-Level-Aktionen. In einer solchen Gruppe wird festgelegt, was auf Server-Level auditiert werden soll. Da Server-Audit-Spezifikationen über eine komplette Instanz eines SQL Servers hinweg konfiguriert werden, ist

es nicht möglich, diese auf eine einzelne Datenbank einzuschränken. [71, S. 390] Der Spezifikation können eine oder mehrere sogenannte „Server-Level-Aktionsgruppen“ hinzugefügt werden. Diese sind vordefinierte Gruppen, um den Server aus Server-Perspektive zu auditieren, und können nicht verändert werden. Es gibt eine Vielzahl an Gruppen, die verwendet werden können. Ein Beispiel dafür wäre die *Successful_Login_Group*, die erfolgreiche Logins auf der Instanz des SQL Servers überwacht. Mittels *state* kann festgelegt werden, ob die Audit-Spezifikation aktiv sein soll oder nicht. [71, S. 391]

- **Datenbank-Audit-Spezifikation** – Diese besteht aus mehreren Gruppen von Datenbank-Level-Aktionen und/oder einzelnen Datenbank-Level-Aktionen. [71, S. 393] Die Gruppen wiederum bestehen aus verschiedenen Datenbank-Level-Aktionen, welche die Überwachung von Aktionen auf Datenbanken, Schemas und Schema-Objekten innerhalb einer Datenbank erlauben. Wie die Gruppen von Server-Level-Aktionen sind auch diese nicht veränderbar. Ein Beispiel dafür ist die Gruppe *Database_Object_Change_Group*, die das Erzeugen, Verändern und Entfernen von Datenbank-Objekten überwacht. [71, S. 394] Im Gegensatz zu der Server-Level-Audit-Spezifikation können auch selbst definierte Datenbank-Level-Audit-Aktionen festgelegt werden. Mit diesen können Aktionen gegen eine Datenbank, ein Schema oder ein Schema-Objekt auditiert werden. Die Ausführung der Statements SELECT, INSERT, DELETE, UPDATE, EXECUTE, RECEIVE und REFERENCES kann überwacht werden. Bei der Definition der Audit-Spezifikation muss angegeben werden, auf welches Securable auditiert werden soll. Ein Securable ist ein Objekt, z. B. Tabelle, View oder Stored Procedure, in der Datenbank. [71, S. 394–395] Auch muss angegeben sein, welche Prinzipale/Entitäten überwacht werden sollen. Beispiele für ein Prinzipal sind DatenbankbenutzerInnen, eine Datenbankrolle oder eine Anwendungsrolle. Hierbei ist es wichtig, zu wissen, dass alle DatenbankbenutzerInnen der Datenbankrolle *public* angehören. [117]

Zusammenfassend lässt sich sagen, dass mit einer Datenbank-Level-Audit-Spezifikation festgelegt wird, welche Arten von Statements, die von einer bestimmten Userin bzw. einem bestimmten User auf eine Tabelle oder eine View ausgeführt werden, überwacht werden sollen. [71, S. 395]

8.4. Umsetzung der Überwachung

8.4.1. Related Work

Im Zuge einer Literaturrecherche konnten keine Arbeiten zum Thema Honeytoken und MSSQL gefunden werden. Arbeiten über die Überwachung einzelner Tabellen existieren sehr wohl. So beschreiben

Woo et al. im Jahr 2005, dass Trigger eingesetzt werden können, um gewisse Aktivitäten zu überwachen. [118, S. 9]

8.4.2. Auditing-Konfiguration

Basierend auf den zur Verfügung stehenden Technologien wurde folgende Lösung für die MSSQL-Datenbank erarbeitet:

- **Überwachung der Salary-Tabelle** – Jeder Zugriff auf diese Tabelle wird mittels der Datenbank-Audit-Spezifikation *MONITOR_SALARY_Employee* überwacht.
- **Überwachung der Honeytokens in der Employee-Tabelle** – In MSSQL gibt es keine Möglichkeit, einzelne Zeilen in einer Tabelle mittels einer Audit-Funktion zu überwachen. Daher wird in der Datenbank-Audit-Spezifikation *MONITOR_SALARY_Employee* auch die Employee-Tabelle zur Überwachung festgelegt.

Die nachfolgenden Konfigurationen könnten anstatt mittels Transact SQL (T-SQL) auch grafisch über die BenutzerInnenoberfläche durchgeführt werden. Listing A.33 zeigt, wie ein Server-Audit-Objekt namens *MyServerAudit* angelegt wird. Dieses schreibt alle Logs in das Verzeichnis *C:\fhstp\Audit*. Das im Skript verwendete GO ist kein T-SQL Statement. Mittels GO wird dem SQL Server Management Studio mitgeteilt, dass alle Befehle seit dem letzten GO oder von Beginn des Skripts an bis zum aktuellen GO ausgeführt werden sollen. [119]

Als nächster Schritt muss die Datenbank-Audit-Spezifikation, wie in Listing A.34 zu sehen, angelegt werden. Das Ausführen von SELECT, INSERT, UPDATE, DELETE, EXECUTE und RECEIVE ON auf die Tabellen *dbo.Salary* und *dbo.Employee* soll damit überwacht werden. Abschließend muss das Server-Audit-Objekt namens *MyServerAudit*, wie in Listing A.35 dargestellt, aktiviert werden.

8.4.3. Remote-Zugriff auf die SQL Server-Datenbank

Über das TDS-Protokoll (Tabular Data Stream) kann von den Anwendungen aus auf den Microsoft-Datenbankserver zugegriffen werden. Das TDS-Protokoll agiert auf der Anwendungsschicht des OSI-Modells und überträgt Anfragen und Antworten zwischen Client und Datenbankserver. [120] Unter Linux steht das unter der GNU LPGPL Lizenz veröffentlichte Projekt FreeTDS zur Verfügung. Dieses bietet Bibliotheken und Programme an, um auf den Microsoft SQL Server zugreifen zu können. [121] Listing A.36 zeigt die Installation und Konfiguration von FreeTDS, um auf die Chinook-Datenbank zugreifen zu können. Anschließend kann mittels des Kommandozeilenprogramms *tsql*, wie in Listing A.37

dargestellt, eine SQL-Abfrage an den Server gesendet werden. In den Parametern wird als Server der zuvor definierte Server angegeben.

8.4.4. Logs ansehen

Im Anschluss an die zuvor durchgeführten Zugriffe können die entsprechenden Audit-Logs mit dem in Listing A.38 dargestellten Befehl angesehen werden.

8.5. Anbindung an die Lure Box

Die Lure Box wird mittels JDBC an die MSSQL-Datenbank angebunden. Die Logs des Microsoft SQL Servers 2016 werden, wie konfiguriert, in einem Verzeichnis im Dateisystem gespeichert. Diese können mittels einer SQL-Funktion ausgelesen werden, für deren Verwendung die SQL Server-Berechtigung *CONTROL SERVER* benötigt wird. [122] Für das Auslesen der Logdaten wird ein Konto namens *auditor* mit der notwendigen Berechtigung, wie in Listing A.39 dargestellt, angelegt.

Damit Logstash mittels JDBC auf die Datenbank zugreifen kann, wird ein entsprechender Treiber benötigt. Dieser kann von der Microsoft-Webseite heruntergeladen [123] und nach dem Download mit den in Listing A.40 gezeigten Befehlen im Verzeichnis */opt/jdbc/mssql* abgelegt werden. Zusätzlich muss ein Datenverzeichnis, wie in Listing A.42 zu sehen, für die neue Logstash-Instanz angelegt werden.

Auf der Lure Box muss die Konfigurationsdatei */etc/logstash/conf.d/mssql_datbase.conf* angelegt werden, deren Inhalt in Listing A.41 ersichtlich ist; danach kann Logstash mittels Supervisor gestartet und überwacht werden. Unter Listing A.43 befindet sich der Inhalt der dafür notwendigen Konfigurationsdatei */etc/supervisor/conf.d/logstash-mssql.conf*. Ist Logstash gestartet, so werden die Logs abgeholt, verarbeitet und in Elasticsearch gespeichert. Damit auch eine E-Mail-Alarmierung erfolgt, muss eine Regel für ElastAlert erstellt werden. Listing A.44 zeigt den Inhalt der Datei */etc/elastalert/rules/monitor_mssql.yaml*.

9. PostgreSQL

PostgreSQL ist ein freies objektrelationales Datenbankmanagementsystem (ORDBMS), das von der PostgreSQL Global Development Group unter einer Open-Source-Lizenz zur Verfügung gestellt wird. Nach mehr als 15 Jahren Entwicklung ist es für seine Verlässlichkeit bekannt und für verschiedene Betriebssysteme wie Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, macOS, Solaris, Tru64) und Windows verfügbar. PostgreSQL ist weitgehend konform mit dem SQL-Standard ANSI-SQL 2008 und unterstützt auch die Speicherung von großen Binärdateien. [124] Gemäß dem Ranking von *db-engines.com* erreichte PostgreSQL im März 2017 den vierten Rang unter den populärsten verfügbaren Datenbanksystemen. [97]

9.1. Installation

Für die Implementierung wurde mittels der Virtualisierungssoftware VirtualBox [73] eine virtuelle Maschine mit folgenden Komponenten installiert:

- **Betriebssystem** – Ubuntu 16.04 64-bit
- **Datenbanksystem** – PostgreSQL 9.5.6

PostgreSQL wurden aus den Paketquellen von Ubuntu 16.04 installiert. [125], [126]

9.2. Import der Chinook-Datenbank

Für die Erstellung und den Import der Datenbank wurden die in Listing A.45 angeführten Befehle ausgeführt. Alle aufgelisteten Befehle starten mit *sudo -u postgres*. Dadurch werden die Befehle im Kontext des Kontos *postgres* ausgeführt, das sich mit der PostgreSQL-Datenbank authentifizieren kann. Nach dem Import der Datenbank muss noch die Salary-Tabelle hinzugefügt werden, auf die keine Person zugreifen darf. Dies wird in Listing A.46 durchgeführt. Darüber hinaus werden ein paar Einträge hinzugefügt.

9.3. Auditing in PostgreSQL 9.5.6

PostgreSQL beinhaltet keine Auditing-Möglichkeiten, die mit denen von Oracle, MS SQL und MongoDB vergleichbar wären. Eine Möglichkeit zur Realisierung von Tabellenüberwachung ist der Einsatz der PostgreSQL-Erweiterung *PGAudit (PostgreSQL Audit Extension)*. [127] Diese bietet zwei verschiedene Auditing-Möglichkeiten an: das sessionbasierte Auditing und das objektbasierte Auditing. Beim sessionbasierten Auditing werden alle Statements geloggt, die von einer bestimmten Userin bzw. einem bestimmten User ausgeführt werden. Beim objektbasierten Auditing können die Statements SELECT, INSERT, UPDATE und DELETE auf eine bestimmte Tabelle geloggt werden. Wesentlicher Nachteil ist, dass PostgreSQL zur Verwendung von *PGAudit* manuell kompiliert und installiert werden muss. Des Weiteren wird *PGAudit* von der Open-Source-Community entwickelt und ist kein offizieller Teil von PostgreSQL. [128]

PostgreSQL besitzt eine eingebaute Bericht- und Logging-Funktionalität. Diese kann so konfiguriert werden, dass alle ausgeführten Statements in eine Datei geloggt werden. Großer Nachteil dieser Methode ist, dass nicht eingeschränkt werden kann, welche Teile der Datenbank genau überwacht werden sollen. Aus diesem Grund kann eine Überwachung der gesamten Datenbank zu Performanceeinbußen führen. [129]

9.4. Umsetzung der Überwachung

9.4.1. Related Work

Wie bei vielen anderen relationalen Datenbanksystemen bietet sich auch in PostgreSQL der Einsatz von Triggern an. Chauhan schlägt den Einsatz eines Triggers vor, um Datenänderungen in eine Historientabelle zu speichern. Der größte Nachteil ist, dass damit keine SELECT Statements geloggt werden können. [130, S. 37]

9.4.2. Auditing-Konfiguration

Basierend auf den zur Verfügung stehenden Technologien wurde entschieden, die PostgreSQL-eigene Methode zum Loggen von Aktivitäten zu wählen. Zwar besteht dabei die Gefahr von Performanceeinbußen, dafür muss die Software nicht manuell kompiliert und keine Drittsoftware installiert werden. Über die Konfigurationsdatei */etc/postgresql/9.5/main/postgresql.conf* kann das Loggen aller ausgeführten Statements aktiviert werden. Der Pfad zu der Konfigurationsdatei lässt sich bei Bedarf über das SQL-Statement *show config_file;* herausfinden. Listing A.47 zeigt die Sektion *ERROR REPORTING*

AND LOGGING in der Konfigurationsdatei von PostgreSQL. Darin wird das Auditing konfiguriert. Zu Beginn wird konfiguriert, dass die Logs im CSV-Format gespeichert werden sollen. CSV hat den großen Vorteil, dass es von Maschinen gut verarbeitet werden kann. [129] Abschließend muss die PostgreSQL-Datenbank, wie in Listing A.48 dargestellt, neu gestartet werden, damit alle Änderungen aktiv werden.

9.4.3. Remote-Zugriff auf die PostgreSQL-Datenbank

Damit das Szenario möglichst realistisch ist, wird ein Remote-Zugriff von der Lure Box auf die Daten in der PostgreSQL-Datenbank durchgeführt. Um dies zu ermöglichen, werden, wie in Listing A.49 zu sehen, dem bereits existierenden Konto *chinookuser* die erforderlichen Berechtigungen eingeräumt. Weiters sind Änderungen in zwei Konfigurationsdateien von PostgreSQL notwendig: In der Datei */etc/postgresql/9.5/main/postgresql.conf* muss konfiguriert werden, dass der Server auch aus dem Netzwerk 192.168.56.0/24 erreichbar ist, und in der Datei */etc/postgresql/9.5/main/pg_hba.conf* müssen die Authentifizierungsmöglichkeiten geändert werden. In Letzterer wird geregelt, welche Clients sich mit der Datenbank verbinden dürfen und welche Authentifizierungsmethode verwendet werden soll. [131] Dem Host 192.168.56.254 muss erlaubt werden, sich über die Methode *md5* zu authentifizieren. Die Durchführung beider Änderungen wird in Listing A.50 beschrieben.

Nachdem diese Konfigurationsänderungen vorgenommen wurden, muss PostgreSQL noch neu gestartet werden. Für den Remote-Zugriff selbst muss ein PostgreSQL-Client auf der Lure Box installiert werden. Listing A.51 zeigt die Installation dieses Clients. Anschließend kann mit dem *chinookuser* eine Verbindung zur Datenbank hergestellt werden und es können verschiedene Abfragen, wie in Listing A.52 zu sehen, durchgeführt werden.

9.4.4. Logs ansehen

Nach den zuvor durchgeführten Zugriffen können die entsprechenden Audit-Logs im Verzeichnis */var/log/postgresql/pg_log* angesehen werden. Da jeder Zugriff auf die Datenbank aufgezeichnet wird, kann der Umfang der Logs in kurzer Zeit sehr rasch anwachsen. Aus diesem Grund wird empfohlen, die Logs zu archivieren und gegebenenfalls zu löschen. Unter Linux würde sich dafür das Programm *logrotate* eignen. [132]

9.5. Anbindung an die Lure Box

Die Logs werden, wie in Unterabschnitt 9.4.4 beschrieben, im CSV-Format unter */var/log/postgresql/pg_log* abgelegt. Daher werden sie mittels Filebeat an die Lure Box übertragen. Filebeat muss auf dem PostgreSQL Server wie in Listing A.2 zu sehen installiert werden. Anschließend kann die Konfiguration über die Datei */etc/filebeat/filebeat.yml* durchgeführt werden. [133] Am Anfang der Datei muss der Pfad für das Input-Verzeichnis angepasst werden. Darüber hinaus wird Filebeat so konfiguriert, dass Logeinträge, die sich über mehrere Zeilen hinweg spannen, richtig verarbeitet werden. Am Ende der Datei müssen der Elasticsearch-Output auskommentiert sowie der Logstash-Output aktiviert und modifiziert werden. Listing A.53 zeigt die erforderlichen Änderungen in der Konfigurationsdatei. Ist die Konfiguration abgeschlossen, so kann die Konfigurationsdatei mit dem Befehl *filebeat.sh -configtest -e* auf ihre Gültigkeit getestet werden. Damit Filebeat die Logdaten erfolgreich auslesen kann, muss das Filebeat-Konto über die entsprechenden Berechtigungen verfügen. Daher wird dieses, wie in Listing A.54 dargestellt, der Gruppe *postgres* hinzugefügt. Weiters müssen die Berechtigungen des Ordners */var/log/postgresql/pg_log/* angepasst werden.

Damit die Logs in Richtung Lure Box gesendet werden, muss Filebeat mittels Supervisor gestartet werden; Listing A.55 zeigt die dafür erforderliche Konfiguration */etc/supervisor/conf.d/filebeat.conf*. Auf der Lure Box muss Logstash für das Entgegennehmen, Verarbeiten und Speichern der Logs konfiguriert werden. Dazu wird die Datei */etc/logstash/conf.d/postgresql_datbase.conf* angelegt und in dieser die in Listing A.56 dargestellte Konfiguration abgelegt. Als Input wird das Plug-in *beats* verwendet, das die Pakete von Filebeat auf Logstash-Seite entgegennimmt. [134]

In der Filtersektion wird mit dem CSV-Filter die von PostgreSQL im CSV-Format abgespeicherte Logzeile in ihre einzelnen Felder aufgeteilt. Die Spaltennamen konnten der PostgreSQL-Webseite entnommen werden. [135] Zusätzlich muss ein Datenverzeichnis, wie in Listing A.57 zu sehen, für die neue Logstash-Instanz angelegt werden. Nachdem die Konfigurationsdatei angelegt wurde, kann Logstash mittels Supervisor gestartet und überwacht werden. Der Inhalt der dafür notwendigen Konfigurationsdatei */etc/supervisor/conf.d/logstash-postgresql.conf* befindet sich unter Listing A.58. Ist Logstash gestartet, werden die Logs abgeholt, verarbeitet und in Elasticsearch gespeichert. Damit auch eine E-Mail-Alarmierung erfolgt, muss eine Regel für ElastAlert erstellt werden. Listing A.59 zeigt den Inhalt der Datei */etc/elastalert/rules/monitor_postgresql.yaml*.

10. MongoDB

MongoDB ist ein Datenbankmanagementsystem, das für das Internet und webbasierte Applikationen erschaffen wurde. Es ist eine dokumentenbasierende NoSQL-Datenbank. Das System wurde für hohen Lese- und Schreibdurchsatz mit einer guten Skalierbarkeit und automatischem Failover geschaffen. MongoDB-Dokumente sind in einem JSON-ähnlichen Format, „BSON“ genannt, enkodiert. BSON, ein vom MongoDB-Team selbst entwickeltes Format, ist aufgrund seiner Leichtgewichtigkeit und Schnelligkeit gut für moderne, objektorientierte Programmiersprachen geeignet. MongoDB ist in der Lage, mittels einer Punktnotation sogar auf verschachtelte BSON-Objekte zuzugreifen; dadurch können Suchaktionen auch in verschachtelten Dokumenten durchgeführt werden. Weitere wichtige Features sind das einfache Clustering und die Replizierung von Daten. [136, S. 53] MongoDB kennt keine Tabellen, Schemas, SQL oder Zeilen. Auch Transaktionen, ACID Compliance, Joins, Foreign Keys und viele andere Features, die in der relationalen Datenbankwelt eingesetzt werden, sind nicht vorhanden. [137, S. 1]

Tabelle 10.1 gibt eine Gegenüberstellung der Terminologie und Konzepte von relationalen Datenbanken und MongoDB.

relationale Datenbank	**MongoDB**
Datenbank	Datenbank
Tabelle	Collection
Zeile	Dokument oder BSON-Dokument
Spalte	Feld
Index	Index
Tabellen-Joins	eingebettete Dokumente und Linking
Primärschlüssel (jede beliebige Spalte oder Kombination von Spalten)	Primärschlüssel (wird automatisch über das _id Feld gesetzt)

Tabelle 10.1.: Gegenüberstellung der Terminologie und Konzepte von relationalen Datenbanken und MongoDB. [65, S. 9]

Gemäß dem Ranking von *db-engines.com* erreichte MongoDB im März 2017 den fünften Rang unter den populärsten verfügbaren Datenbanksystemen. [97]

10.1. Installation

Für die Implementierung wurde mittels der Virtualisierungssoftware VirtualBox [73] eine virtuelle Maschine mit folgenden Komponenten installiert:

- **Betriebssystem** – Ubuntu 16.04
- **Datenbanksystem** – MongoDB Enterprise 3.4

MongoDB stellt eine Anleitung zur Verfügung, wie MongoDB Enterprise 3.4 in wenigen Schritten aus den Paketquellen installiert werden kann. Nach der Installation sind die in Tabelle 10.2 dargestellten Pakete vorhanden: [138]

Paketname	**Inhalt**
mongodb-enterprise-server	Beinhaltet den *mongod* Daemon und damit verbundene Konfigurationen und *init*-Skripte. Der *mongod* ist der Daemon für eine einzelne Datenbank.
mongodb-enterprise-mongos	Beinhaltet den *mongos* Daemon. Dieser kommt zum Einsatz, wenn ein MongoDB-Cluster verwendet wird.
mongodb-enterprise-shell	Beinhaltet die MongoDB Shell, den Client, mit dem auf Datenbanken zugegriffen werden kann.
mongodb-enterprise-tools	Beinhaltet viele MongoDB-Tools wie *mongoimport bsondump*, *mongodump*, *mongoexport*, *mongofiles*, *mongooplog*, *mongoperf*, *mongorestore*, *mongostat* und *mongotop*.

Tabelle 10.2.: Pakete, die mittels des Metapackages *mongodb-enterprise* installiert werden.

10.2. Importieren einer Datenbank

Da die Chinook-Datenbank nicht für NoSQL-Datenbanken vorhanden ist, wurde eine von MongoDB zur Verfügung gestellte Beispieldatenbank verwendet. [139] Der Import der Beispieldaten wird in Listing A.60 dargestellt. Zuerst wird die Datenbank als JSON-Datei heruntergeladen und anschließend mittels *mongoimport* importiert. Würde die Datenbank bereits existieren, so würde die existierende Version

davor gelöscht werden. Nach dem Import sind eine Datenbank namens *fhstp* und eine Collection namens *restaurants* vorhanden. In Letzterer befinden sich 25 359 Dokumente. Zusätzlich muss eine Salary Collection hinzugefügt werden, auf die keine Person zugreifen darf; dies wird in Listing A.61 durchgeführt. Darüber hinaus werden weitere Einträge in die Salary Collection hinzugefügt. Die einzelnen Dokumente werden mit einem Array an die MongoDB Shell übergeben.

10.3. Auditing in MongoDB Enterprise 3.4

Auditing-Funktionalitäten sind derzeit lediglich in den Enterprise-Editionen von MongoDB enthalten. [140] Als Alternative könnte zum Percona Server für MongoDB gegriffen werden. Dieser dient laut Herstellerangaben als kostenloser Ersatz zu MongoDB. Der Unterschied besteht darin, dass Enterprise-Funktionen wie Auditing inkludiert sind. [141]

MongoDB Enterprise bietet Auditing-Möglichkeiten für die Instanzen *mongod* und *mongos*. Ist Auditing einmal aktiviert, so können die folgenden Operationen auditiert werden: [140], [142, S. 3]

- **Schema (DDL)** – Operationen wie CREATE, DROP und UPDATE von Collection-Schemas, Datenbanken und UserInnen
- **Operationen, die den Cluster betreffen**
- **Authentifizierung und Autorisierung** – Logins von UserInnen und Kommandos, die ohne Berechtigung ausgeführt werden
- **CRUD Operationen** – (dafür muss der Parameter *auditAuthorizationSuccess* auf *true* gesetzt werden) Dadurch wird überwacht, wenn bei einer Collection Daten eingefügt, gelesen, geändert oder gelöscht werden.

Das Auditing-System schreibt jedes Event in einen Puffer, der von MongoDB in periodischen Abständen auf die Festplatte geschrieben wird. MongoDB garantiert, dass bereits alle vorhergegangenen Events ebenfalls schon gespeichert wurden, wenn ein Event auf die Festplatte geschrieben wird. [140] Die dabei entstehenden Events können in einer JSON- oder BSON-Datei abgespeichert werden. Unter Linux steht zusätzlich die Möglichkeit der Verwendung von *syslog* zur Verfügung.

Damit MongoDB Aktivitäten auditiert, müssen gewisse Parameter beim Start des *mongod* Daemons angegeben werden oder in der *mongod* Konfigurationsdatei festgelegt werden. [143] Ist das Auditing aktiv,

so wird standardmäßig alles auditiert, wozu MongoDB in der Lage ist. Mithilfe von Audit-Filtern kann dies eingeschränkt werden und es lässt sich genauer bestimmen, was auditiert werden soll.

Als Alternative zu der Auditing-Funktionalität bietet MongoDB auch einen sogenannten „Database Profiler“ sowie ein Tool namens *mongoreplay* an. Mit dem Profiler können Daten über READ, UPDATE und andere von BenutzerInnen ausgeführte Kommandos gesammelt werden. Der Profiler ist standardmäßig deaktiviert. Nachteil des Profilers ist, dass nicht eingeschränkt werden kann, welche Collections genau überwacht werden sollen. [144] Mittels *mongoreplay* können die Kommandos aufgezeichnet werden, die an eine MongoDB-Instanz gesendet werden. [145]

10.4. Umsetzung der Überwachung

10.4.1. Related Work

Prabhakaran et al. sahen sich im Jahr 2014 die Logging-Möglichkeiten an, die von MongoDB zur Verfügung gestellt wurden. Die in der vorliegenden Arbeit eingesetzte Auditing-Funktionalität findet darin keine Erwähnung, da es dieses Feature zum damaligen Zeitpunkt vermutlich noch nicht gab. [136, S. 55] Auch schlugen sie ein System vor, bei dem die von MongoDB generierten Logs mittels Logstash normalisiert werden und in einer Elasticsearch-Datenbank abgelegt werden sollen. [136, S. 57]

10.4.2. Auditing-Konfiguration

Basierend auf den zur Verfügung stehenden Technologien wurde folgende Lösung für die MongoDB-Datenbank erarbeitet:

- **Überwachung der Salary Collection** – Jeglicher Zugriff auf diese Collection wird mittels der Audit-Funktion von MongoDB überwacht.
- **Überwachung der Honeytokens in der Restaurants Collection** – Da es dem Autor dieser Arbeit nicht möglich war, die Audit-Funktion auf einzelne Dokumente einzuschränken, wird diese Collection vollständig überwacht.

Listing A.62 zeigt, wie die Konfigurationsdatei von MongoDB */etc/mongod.conf* konfiguriert werden muss. Diese ist eine Konfigurationsdatei, die sich an den YAML-Syntax hält. In der Enterprise-only-Sektion in der Konfigurationsdatei wird das Auditing konfiguriert.

10.4.3. Remote-Zugriff auf die MongoDB-Datenbank

Damit das Szenario möglichst realistisch ist, wird ein Remote-Zugriff von der Lure Box auf die Daten in der MongoDB-Datenbank durchgeführt. Für den Remote-Zugriff auf die MongoDB-Datenbank muss ein neues Konto, wie in Listing A.63 ersichtlich, angelegt und anschließend die Konfigurationsdatei von MongoDB angepasst werden. In der Datei */etc/mongod.conf* musste konfiguriert werden, dass der Server auch aus dem Netzwerk 192.168.56.0/24 erreichbar ist und BenutzerInnen sich mit der Datenbank verbinden können. Die dafür erforderlichen Konfigurationsänderungen werden in Listing A.64 gezeigt. [146] Nachdem diese Konfigurationsänderungen vorgenommen wurden, muss MongoDB noch neu gestartet werden. Für den Remote-Zugriff selbst muss der MongoDB Enterprise Shell Client installiert werden, über den dann Abfragen abgesetzt werden können. Listing A.65 zeigt die Installation der Shell. Mit den in Listing A.66 dargestellten Befehlen kann eine Verbindung mit der Chinook-Datenbank auf dem Server 192.168.56.4 hergestellt werden und es können verschiedene Abfragen durchgeführt werden.

10.4.4. Logs ansehen

Nach den zuvor durchgeführten Zugriffen können die entsprechenden Audit-Logs in der Datei */var/lib/-mongodb/auditLog.json* angesehen werden.

10.5. Anbindung an die Lure Box

Die Logs werden im JSON-Format unter */var/lib/mongodb/auditLog.json* abgelegt. Daher werden sie mittels Filebeat an die Lure Box übertragen. Wie in Listing A.2 zu sehen, muss Filebeat auf dem MongoDB-Server installiert werden. Anschließend kann die Konfiguration über die Datei */etc/filebeat/-filebeat.yml* durchgeführt werden. [133] Am Anfang der Datei muss der Pfad für das Input-Verzeichnis angepasst werden, am Ende der Datei muss der Elasticsearch-Output auskommentiert und der Logstash-Output aktiviert und modifiziert werden. Listing A.67 zeigt die erforderlichen Änderungen in der Konfigurationsdatei. Ist die Konfiguration abgeschlossen, kann die Konfigurationsdatei mit dem Befehl *filebeat.sh -configtest -e* auf ihre Gültigkeit getestet werden. Damit die Logs in Richtung Lure Box gesendet werden, muss Filebeat mittels Supervisor gestartet werden. Listing A.68 zeigt die dafür erforderliche Konfiguration der Datei */etc/supervisor/conf.d/filebeat.conf*.

Auf der Lure Box muss Logstash für das Entgegennehmen, Verarbeiten und Speichern der Logs konfiguriert werden. Dazu wird die Datei */etc/logstash/conf.d/mongodb_datbase.conf* angelegt. In dieser wird die in Listing A.69 dargestellte Konfiguration abgelegt. Als Input wird das Plug-in *beats* ver-

wendet, das die Pakete von Filebeat auf der Logstash-Seite entgegennimmt. [134] In der Filtersektion wird mit dem JSON-Filter die von MongoDB im JSON-Format abgespeicherte Logzeile in ihre einzelnen Felder aufgeteilt. Die Bedeutung der einzelnen Spalten konnten der MongoDB-Webseite entnommen werden. [147] Zusätzlich muss ein Datenverzeichnis, wie in Listing A.70 zu sehen, für die neue Logstash-Instanz angelegt werden. Nachdem die Konfigurationsdatei angelegt wurde, kann Logstash mittels Supervisor gestartet und überwacht werden. Den Inhalt der dafür notwendigen Konfigurationsdatei */etc/supervisor/conf.d/logstash-mongodb.conf* befindet sich unter Listing A.71. Ist Logstash gestartet, werden die Logs abgeholt, verarbeitet und in Elasticsearch gespeichert. Damit auch eine E-Mail-Alarmierung erfolgt, muss eine Regel für ElastAlert erstellt werden; Listing A.72 zeigt den Inhalt der Datei */etc/elastalert/rules/monitor_mongodb.yaml*.

11. Postfix E-Mail-Server

Postfix ist ein Mailserver, der von Wietse Venema entwickelt und im Dezember 1998 erstmals als Open-Source-Software veröffentlicht wurde. Es sollte eine Alternative zu dem weitverbreiteten Sendmail sein. [148, S. 1] Postfix nimmt die im SMTP-Protokoll definierte Rolle des MTA (Mail Transfer Agent) ein und kümmert sich daher um die Zustellung von Nachrichten zwischen zwei Servern sowie innerhalb eines lokalen Systems. [148, S. 5] Für das Betreiben von Postfix sind UNIX-ähnliche Systeme wie AIX, BSD, HP-UX, Linux, MacOS X oder Solaris erforderlich. [149]

11.1. Installation

Für die Implementierung wurde mittels der Virtualisierungssoftware VirtualBox [73] eine virtuelle Maschine mit folgenden Komponenten installiert:

- **Betriebssystem** – Ubuntu 16.04
- **Mailserver** – Postfix 3.1.0-3

Postfix wurden aus den Paketquellen von Ubuntu 16.04 installiert. [150] Die Installation und anschließende Konfiguration wird in Listing A.73 dargestellt. Der Mailserver wird so konfiguriert, dass er sich für Mails mit *chinookcorp.com* in der Empfangsadresse zuständig fühlt und diese entgegennimmt. [151]

11.2. Erstellen der überwachten E-Mail-Konten

Damit AngreiferInnen auch Nachrichten an die Honeytoken-E-Mail-Adressen senden können, müssen diese am E-Mail-Server angelegt werden. Als Honeytoken-MitarbeiterInnen wurden Margaret Park und Steve Johnson ausgewählt. Weitere Informationen über die entsprechenden MitarbeiterInnen sind in Tabelle 11.1 ersichtlich.

EmployeeId	Nachname	Vorname	Geburtsdatum	Stadt	E-Mail-Adresse
4	Park	Margaret	19.09.1947	Calgary	margaret@chinookcorp.com
5	Johnson	Steve	03.03.1965	Calgary	steve@chinookcorp.com

Tabelle 11.1.: MitarbeiterInnen aus der Chinook-Datenbank, die überwacht werden. [62]

Listing A.74 zeigt, wie die betreffenden BenutzerInnen angelegt werden.

11.3. Auditing in Postfix 3.1.0

Postfix läuft im Hintergrund und loggt Probleme und normale Aktivitäten mittels *syslog*. Der *syslogd* Prozess sortiert Events nach Klassen und Kritikalität und hängt diese einer Logdatei an. Standardmäßig ist dies die Datei */var/log/mail.log*. [151] Wird von Postfix eine E-Mail empfangen, so werden auch AbsenderIn und EmpfängerIn geloggt.

11.4. Umsetzung der Überwachung

Für das Auditing mussten keine Konfigurationen vorgenommen werden.

11.4.1. Versenden von E-Mails von der Lure Box aus

Mittels des Kommandozeilenprogramms *swaks* können Nachrichten an die MitarbeiterInnen Margaret Park und Steve Johnson versendet werden. [152] Die Installation des Programms sowie das Versenden der Nachrichten wird in Listing A.75 dargestellt. Die ersten beiden Mails mit dem Absender newsletter@chinookcorp.com sollen keinen Alarm generieren, die letzten beiden Nachrichten jedoch schon.

11.4.2. Logs ansehen

Nach dem Versand der Nachrichten können die entsprechenden Logs in der Datei */var/log/mail.log* gefunden werden. Listing 11.1 zeigt einen Auszug davon.

```
1 # message from newsletter@chinookcorp.com to steve@chinookcorp.com
2 postfix/qmgr[20871]: 105851A7A: from=<newsletter@chinookcorp.com>, size=424,
   ↪ nrcpt=1 (queue active)
3 postfix/local[20955]: 105851A7A: to=<steve@chinookcorp.com>, relay=local,
   ↪ delay=0.06, delays=0.03/0/0/0.03, dsn=2.0.0, status=sent (delivered to
   ↪ mailbox)
4 postfix/qmgr[20871]: 105851A7A: removed
5
6 # message from attack@insider.com to margaret@chinookcorp.com
7 postfix/qmgr[20871]: 287EA1A7A: from=<attack@insider.com>, size=422, nrcpt=1 (queue
   ↪ active)
8 postfix/local[20955]: 287EA1A7A: to=<margaret@chinookcorp.com>, relay=local,
   ↪ delay=0.06, delays=0.03/0/0/0.03, dsn=2.0.0, status=sent (delivered to
   ↪ mailbox)
9 postfix/qmgr[20871]: 287EA1A7A: removed
```

Listing 11.1: Auszug aus den Postfix-Logs.

11.5. Anbindung an die Lure Box

Die Logs werden, wie in Abschnitt 11.3 beschrieben, unter */var/log/mail.log* abgelegt. Daher werden sie mittels Filebeat an die Lure Box übertragen. Filebeat muss auf dem Postfix-Server, wie in Listing A.2 zu sehen, installiert werden; anschließend kann die Konfiguration über die Datei */etc/filebeat/filebeat.yml* durchgeführt werden. [133] Am Anfang der Datei muss der Pfad für das Input-Verzeichnis angepasst werden, am Ende der Datei muss der Elasticsearch-Output auskommentiert sowie der Logstash-Output aktiviert und modifiziert werden. Listing A.76 zeigt die erforderlichen Änderungen in der Konfigurationsdatei. Ist die Konfiguration abgeschlossen, kann die Konfigurationsdatei mit dem Befehl *filebeat.sh -configtest -e* auf ihre Gültigkeit getestet werden.

Damit Filebeat die Logdaten erfolgreich auslesen kann, muss das Filebeat-Konto über die entsprechenden Berechtigungen verfügen; daher werden an dieses die benötigten Rechte wie in Listing A.77 dargestellt vergeben. Damit die Logs in Richtung Lure Box gesendet werden, muss Filebeat mittels Supervisor gestartet werden; Listing A.78 zeigt die dafür erforderliche Konfiguration der Datei */etc/supervisor/conf.d/filebeat.conf*.

Auf der Lure Box muss Logstash für das Entgegennehmen, Verarbeiten und Speichern der Logs konfiguriert werden. Dazu wird die Datei */etc/logstash/conf.d/postfix.conf* angelegt, in der die in Listing A.79 dargestellte Konfiguration abgelegt wird. Als Input wird das Plug-in *beats* verwendet, das die Pakete von

Filebeat auf Logstash-Seite entgegennimmt. [134] In der Filtersektion wurde auf öffentlich verfügbare Grok Patterns zurückgegriffen. [153], [154] Ein Problem der Postfix-Logs besteht darin, dass die Quelle und das Ziel einer E-Mail-Nachricht nur über mehrere Logzeilen hinweg abspeichert werden. Aus diesem Grund muss das Logstash Aggregate Plug-in eingesetzt werden. Dieses erlaubt es, Informationen, die über mehrere Logzeilen hinweg verteilt sind, zu einer einzelnen Zeile zu aggregieren. Damit das Plug-in verwendet werden kann, muss es wie in Listing A.80 dargestellt installiert werden. [155]

Das Plug-in benötigt eine *task_id*, um Events (Loglines), die zu einer Aktion gehören, korrelieren zu können. Am Beginn einer Aktion wird vom Filter eine *map*-Variable erstellt, die an die *task_id* gebunden wird. Für jedes Event (Logline) kann ein Code ausgeführt werden, bei dem auf die *event*-Variable oder auf die *map*-Variable zugegriffen wird. Die *event*-Variable enthält die aktuell verarbeitete Logline. Wird das letzte Event einer Aktion empfangen, so kann ein Code ausgeführt werden, bei dem beispielsweise der Inhalt der *map*-Variablen dem Event hinzugefügt wird. Das finale Event wird mit der Konfiguration *end_of_task => true* gekennzeichnet. Tritt dies auf, so wird die *map*-Variable für die *task_id* verworfen. Zusätzlich kann ein Timeout definiert werden, nach dessen Ablauf ein *map*-Objekt veröffentlicht werden soll, ohne dass alle erforderlichen Teile empfangen wurden. [155] Listing 11.2 zeigt die Anwendung des Aggregate Plug-ins. Zu Beginn wird überprüft, ob das Feld *postix_from*, das den bzw. die AbsenderIn einer E-Mail enthält, vorhanden ist. Ist dies der Fall, so kommt das Aggregate Plug-in zum Einsatz. Die *postfix_queueid* wird als *task_id* verwendet. Die Absendeadresse wird in der *map*-Variablen abgelegt. Wird das abschließende Event empfangen, in dem die Empfangsadresse gespeichert ist, so wird dem Event die zuvor gespeicherte Absendeadresse mithilfe der *map*-Variable angehängt. [155]

```
# perform aggregation of the postix_from and the postfix_to event
if [postfix_from] {
        aggregate {
                task_id => "%{postfix_queueid}"
                code => "map['from'] = event.get('postfix_from')"
                map_action => "create"
        }
}
if [postfix_to] {
        aggregate {
                task_id => "%{postfix_queueid}"
                code => "event.set('postfix_from',map['from'])"
                map_action => "update"
                end_of_task => true
                push_map_as_event_on_timeout => true
                timeout => 600 # 10 minutes timeout
        }
}
```

Listing 11.2: Anwendung des Logstash Aggregate Plug-ins für Postfix-Logs.

Bevor Logstash gestartet werden kann, muss ein Daten-Verzeichnis, wie in Listing A.81 zu sehen, angelegt werden. Nachdem die Konfigurationsdatei angelegt wurde, kann Logstash mittels Supervisor gestartet und überwacht werden. Unter Listing A.82 befindet sich der Inhalt der dafür notwendigen Konfigurationsdatei */etc/supervisor/conf.d/logstash-postfix.conf*. Es ist wichtig, dass Logstash mit dem Argument *-w 1* gestartet wird. Dies teilt Logstash mit, dass nur ein einziger Worker verwendet wird. Ansonsten könnte es sein, dass das Plug-in nicht wie gewünscht funktioniert. Ist Logstash gestartet, werden die Logs abgeholt, verarbeitet und in Elasticsearch gespeichert. Damit auch eine E-Mail-Alarmierung erfolgt, muss eine Regel für ElastAlert erstellt werden. Listing A.83 zeigt den Inhalt der Datei */etc/elastalert/rules/-monitor_postfix.yaml*.

12. Visualisierung

Dieses Kapitel stellt verschiedene Möglichkeiten der graphischen Auswertung und Visualisierung der erzeugten Logdaten und Alarme vor.

Greifen AngreiferInnen auf Honeytokens zu, so wird eine Alarm-E-Mail verschickt. Listing 12.1 zeigt eine E-Mail, die eine Minute nach dem Versand einer E-Mail an die Honeytoken-E-Mail-Adresse *margaret@chinookcorp.com* empfangen wurde.

```
From: elastalert@fhstp.ac.at
To: incident@fhstp.ac.at

An external mail to an internal address margaret@chinookcorp.com was received.
The internal address is only allowed to receive messages from internal senders.
Mails from external addresses indicate data loss.
The sender address was: attack@insider.com
Connect to the Kibana webinterface http://192.168.56.254:5601 to receive more
 ↪  information.
```

Listing 12.1: Alarmierungs-E-Mail, die unter der Adresse *incident@fhstp.ac.at* empfangen wurde.

In der Nachricht wird auf das Webinterface von Kibana verwiesen. Darin können die entsprechenden Logs aller angebundenen Systeme angesehen werden. Die Elasticsearch-Indexe müssen davor in Kibana hinzugefügt werden. Dazu Kibana öffnen → *Management* → *Index Patterns* und folgende Indexe hinzufügen:

- **audit_*** – Enthält alle empfangenen Audit-Logs von allen Systemen.
- **audit_database_mssql-*** – Enthält die Logs des Microsoft SQL Servers.
- **audit_datbase_oracle-*** – Enthält die Logs der Oracle-Datenbank.
- **audit_datbase_postgresql-*** – Enthält die Logs der PostgreSQL-Datenbank.
- **audit_postfix-*** – Enthält die Logs von Postfix.

Die Namen der Indexe wurden im Output-Teil von jeder Logstash-Konfiguration definiert. Sind die Indexe hinzugefügt, so können die Logs über das Menü *Discover* angesehen werden. Abbildung 12.1 zeigt eine Übersicht über mehrere von Oracle generierte Logs.

Oracle-Honeytoken

Time	src.ip	audit.object.schema	audit.object.name	log_source.host
May 10th 2017, 16:01:28.653	192.168.56.254	CHINOOK	SALARY	192.168.56.1
May 10th 2017, 16:01:28.653	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 24th 2017, 17:05:45.074	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 24th 2017, 17:05:45.074	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 24th 2017, 17:04:24.304	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 24th 2017, 17:04:24.304	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 12th 2017, 19:05:46.688	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 12th 2017, 19:05:46.688	192.168.56.254	CHINOOK	SALARY	192.168.56.1
April 12th 2017, 18:59:09.297	192.168.56.254	CHINOOK	SALARY	192.168.56.1

Abbildung 12.1.: Darstellung von Oracle-Logs in Kibana.

Es können auch einzelne Events detailliert angesehen werden. Abbildung 12.2 zeigt ein von Postfix generiertes Event, das einen Alert mit ElastAlert ausgelöst hat.

Table JSON

@timestamp	May 11th 2017, 17:11:03.000
@version	1
_id	AVv4EVPpfFfqdjNAIZja
_index	audit_postfix-2017.05.11
_score	-
_type	logs
audit.file.name	/var/log/mail.log
audit.file.offset	13,716
audit.postfix.component	postfix/local
audit.postfix.from	attack@insider.com
audit.postfix.queue_id	288991A7A
audit.postfix.relay	local
audit.postfix.status	sent
audit.postfix.to	margaret@chinookcorp.com
log_source.host	192.168.56.5
log_source.product	Postfix
log_source.received_at	May 11th 2017, 17:12:13.072
tags	beats_input_codec_plain_applied, _grok_postfix_success, create_alert

Abbildung 12.2.: Darstellung eines einzelnen, von Postfix generierten Events in Kibana.

Kibana bietet viele weitere Möglichkeiten, um Logs zu filtern und zu visualisieren. Es können auch sogenannte „Dashboards“ gebaut werden, in denen bereits zuvor erstellte Visualisierungen und Suchaktionen platziert werden, die dann laufend aktualisiert werden können. Dadurch ist ein visuelles Monitoring möglich.

13. Zusammenfassung und Ausblick

Ziel der vorliegenden Arbeit war es, die Lure Box vorzustellen und einen Einblick in das damit verbundene Konzept zu geben. Dieses sieht den Einsatz von Honeytokens, eine spezielle Form von Honeypots, in möglichst vielen verschiedenen Systemen vor. Honeytokens sind digitale oder analoge Ressourcen, deren Wert darin liegt, von jemandem unautorisiert verwendet zu werden. [12] Das können beispielsweise Tabellen in einer Datenbank sein oder Dateien auf einem Server, auf die niemand zugreifen sollte. Findet dennoch ein Zugriff statt, so wird dieser beispielsweise in einem Logeintrag festgehalten. Vorteil der Honeytokens ist, dass diese sehr kostengünstig und effizient an verschiedenen Orten in einem Unternehmen platziert werden können. Diese Arbeit stellte deren Vor- und Nachteile sowie verschiedene Einsatzszenarien in einem Unternehmensnetzwerk vor.

Die Lure Box selbst ist eine auf Debian basierende virtuelle Maschine, die den kostenlos einsetzbaren Elastic Stack [156] der Firma Elasticsearch installiert hat. Mit diesem können Logdaten von verschiedenen Systemen empfangen, verarbeitet und in einer Datenbank abgespeichert werden. Mittels des ebenfalls kostenlos verfügbaren ElastAlert [61] können Alarme generiert werden, wenn Logdaten eintreffen, die zuvor definierten Kriterien entsprechen. Um die Verwendung der Lure Box sowie deren Möglichkeiten besser zu veranschaulichen, wurde ein umfassendes Szenario erstellt. Es wurden virtuelle Maschinen mit aktuellen Versionen der Datenbanksysteme von Oracle, Microsoft, PostgreSQL und MongoDB installiert und konfiguriert. Alle Datenbanksysteme wurden mit Beispieldaten und einer Salary-Tabelle/Collection befüllt. Diese dient als Honeytoken und daher wird jeder Zugriff darauf protokolliert. In einer anderen Tabelle/Collection befinden sich die Stammdaten von MitarbeiterInnen eines Unternehmens. Davon wurden zwei zu sogenannten Fake-MitarbeiterInnen erklärt, die in einem echten Unternehmen gar nicht existieren würden. Aus diesem Grund gibt es auch keine Veranlassung, warum eine externe Person E-Mail-Nachrichten an die Adressen der beiden MitarbeiterInnen senden sollte. Wird dennoch eine E-Mail empfangen, kann das Unternehmen davon ausgehen, dass es Opfer einer Cyberattacke wurde oder dass von internen MitarbeiterInnen Daten unerlaubt weitergegeben wurden. Damit auch die Verwendung der Honeytoken-E-Mail-Adressen simuliert werden kann, wurde ein Postfix-E-Mail-Server eingerichtet. Auf diesem wurde, wie auf allen Datenbanksystemen auch, Auditing aktiviert

und konfiguriert.

Die vorliegende Arbeit zielte darauf ab, einen möglichst guten Überblick über die unterschiedlichen Auditing-Möglichkeiten der vier eingesetzten Datenbanksysteme zu geben. Dabei stellte sich heraus, dass die beiden kommerziellen Hersteller Oracle und Microsoft die umfassendsten Auditierungsmöglichkeiten bieten. Bei diesen konnte sehr klar definiert werden, welche Aktionen überwacht werden sollen. Bei MongoDB war dies nicht mehr ganz so detailliert möglich. PostgreSQL hingegen konnte lediglich dahingehend konfiguriert werden, dass sämtliche Zugriffe auf die Datenbank aufgezeichnet werden, was zu hohen Performanceeinbußen führen kann. Neben MongoDB sollte ursprünglich auch eine weitere NoSQL-Datenbank Teil der Arbeit sein, weshalb die beiden NoSQL-Datenbanken Apache Cassandra [157] und Neo4j [158] für die Verwendung in dieser Arbeit geprüft wurden. Die getestete Apache Cassandra Version 3.10 unterstützt jedoch derzeit kein Auditing, und in der Apache Cassandra Dokumentation waren keine Angaben über Auditing zu finden. [159] Auch bei der Neo4J-Datenbank konnten keine Hinweise zu Auditing-Funktionalitäten gefunden werden. [160] Dies zeigt, dass bei den etablierten relationalen Datenbanksystemen Sicherheit und Auditing weiter fortgeschritten sind und dass der Fokus bei den neueren NoSQL-Datenbanken auf andere Dinge gelegt wird.

Greifen AngreiferInnen nun auf die Salary-Tabellen/Collections in den Datenbanken zu, so generiert die Datenbank entsprechende Logdaten, die an die Lure Box übertragen und dort verarbeitet sowie dauerhaft abgespeichert werden. Das Programm ElastAlert überwacht mit zuvor definierten Regeln die abgespeicherten Daten. Wird ein Zugriff auf einen Honeytoken festgestellt, so wird ein Alarm per E-Mail versendet. AnwenderInnen können sich dann auf ein Webinterface verbinden und dort die erzeugten Logs von allen angebundenen Systemen einer detaillierten Prüfung unterziehen.

Die Lure Box zeigt, dass mit verhältnismäßig wenig Budget die IT-Sicherheit in einem Unternehmen signifikant erhöht werden kann. Honeytokens sollten dabei jedoch nicht als alleiniges Wundermittel verstanden, sondern immer als Teil eines umfassenden Sicherheitskonzeptes angesehen werden. Sie sind eine Warnung, dass sich jemand Zugriff auf Dinge verschafft hat, auf die nicht zugegriffen werden sollte. Je früher das Sicherheitsteam in einem Unternehmen über solche Vorgänge Bescheid weiß, desto eher können Nachforschungen angestellt und Gegenmaßnahmen getroffen werden.

Als nächster Schritt sollte eine Webseite für die Lure Box erstellt werden. Mittels dieser können die Box und das dahinterliegende Konzept einer breiten Öffentlichkeit zugänglich gemacht werden. Es könnten

virtuelle Maschinen, Docker-Container oder sogar Pakete für bestimmte Distributionen gebaut werden. Interessierte AnwenderInnen könnten sich diese dann herunterladen und austesten. Mithilfe der Online-Community könnte eine stetig steigende Sammlung von möglichen Einsatzszenarien für Honeytokens erstellt werden. Dabei könnte detailliert erklärt werden, wie die Honeytokens platziert werden und wie das Auditing der jeweiligen Systeme konfiguriert werden muss. Eine Sammlung mit Anleitungen für die richtige Anbindung und Verarbeitung von bestimmten Logquellen könnte ebenfalls erstellt werden.

Für ein Unternehmen ist es wichtig, zu wissen, wie sich das Aktivieren von Auditing auf die Performance der darunterliegenden Systeme auswirkt. Daher sollten umfassende Tests unter möglichst realen Bedingungen durchgeführt werden. Auch die Möglichkeiten zur sicheren und zuverlässigen Übertragung von Logdaten sollten genauer untersucht werden, denn nur so kann sichergestellt werden, dass AngreiferInnen nicht gefälschte Logs mittels UDP an die Lure Box senden. Für die bei der Lure Box angekommenen Logs sollte ein einheitliches Normalisierungskonzept erstellt werden. Dieses würde das Schreiben von Regeln und die Analyse von Logdaten über verschiedene Logquellen hinweg erleichtern.

Ein wichtiger Punkt ist auch die Steigerung der Qualität von Honeytokens und deren Platzierung in realen Daten. Je echter diese nach außen erscheinen und je besser sie verteilt sind, desto eher werden AngreiferInnen darauf zugreifen. Honeytokens könnten mit einem Tool wie Honeygen generiert werden. Dieses lernt mittels Echtdaten und kann, basierend auf diesen, Fake-Datensätze erzeugen. [35] Ein weiterer Punkt wäre, nach Möglichkeiten für die Kommunikation mit IDS-Systemen zu suchen. Diese könnten, sobald ein Alarm generiert wurde, jeglichen Netzwerkverkehr aufzeichnen, der zu und von dem betroffenen Host stattfindet.

In einer sehr fortgeschrittenen Version könnte die Lure Box so weit automatisiert werden, dass AnwenderInnen nur geringe Kenntnisse über Honeytokens und die Anbindung der Systeme an die Lure Box benötigen. Über einen Installations-Wizard könnten gewünschte Zielsysteme ausgewählt werden, wo Honeytokens platziert werden sollen. Sind die Adresse und die Zugangsdaten zu dem System eingegeben, kann automatisiert eine Verbindung hergestellt werden. AnwenderInnen könnten dann beispielsweise Daten bestimmen, aus denen gelernt werden soll, und auch ein Ziel angeben, wo die künstlich erzeugten Daten hingeschrieben werden sollen. Sind die Daten generiert und platziert, können auch automatisch das dafür notwendige Auditing und die richtige Alarmierung konfiguriert werden.

A. Quellcode-Sammlung

Hier im Anhang befindet sich eine Sammlung der getätigten Konfigurationen und des während der Arbeit erzeugten Quellcodes.

A.1. Allgemein

A.1.1. Elasticsearch-Paketquellen zum System hinzufügen

```
# 1. download and install the public key from Elasticsearch
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
# 2. install the apt-transport-https packages for Debian
sudo apt-get install apt-transport-https
# 3. add the Elastic repository to the system
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a
 ↪  /etc/apt/sources.list.d/elastic-5.x.list
```

Listing A.1: Hinzufügen der Elasticsearch-Paketquellen zu einem Debian-System. [161]

A.1.2. Installation von Filebeat

```
# update package repository and install filebeat
sudo apt-get update && sudo apt-get install filebeat

# add a filebeat user to the system and assign necessary privileges to access
 ↪  directories
useradd -r filebeat
mkdir /var/log/filebeat
mkdir /var/lib/filebeat
chown -R filebeat:filebeat /etc/filebeat/
chown -R filebeat:filebeat /var/lib/filebeat/
```

Listing A.2: Installation von Filebeat über die Paketquellen. [76]

A.2. Lure Box

A.2.1. Installation von Elasticsearch

```
# download Oracle JDK Version 8
wget http://download.oracle.com/otn-pub/java/jdk/.../jdk-8u121-linux-x64.tar.gz
# extract Java into the /opt/jdk directory
mkdir /opt/jdk
tar -zxf jdk-8u121-linux-x64.tar.gz -C /opt/jdk
# configure Oracle JDK as default JVM
update-alternatives --install /usr/bin/java java /opt/jdk/jdk1.8.0_121/bin/java 100
update-alternatives --install /usr/bin/javac javac /opt/jdk/jdk1.8.0_121/bin/javac
 ↪  100

# install Elasticsearch from the package repository
sudo apt-get update && sudo apt-get install elasticsearch
# tell the system to boot Elasticsearch automatically after reboot
systemctl enable elasticsearch.service
# start Elasticsearch
service elasticsearch start
```

Listing A.3: Installation von Elasticsearch und allen benötigten Abhängigkeiten. [161], [162]

A.2.2. Installation von Cerebro

```
# add a cerebro user to the system
useradd -r cerebro
# download current version from https://github.com/lmenezes/cerebro/releases herunterladen
wget https://github.com/lmenezes/cerebro/releases/download/v0.6.4/cerebro-0.6.4.tgz
# create a directory where cerebro shall be extracted to. The user cerebro shall become owner of the directory.
mkdir /opt/cerebro
tar -xvf cerebro-0.6.4.tgz -C /opt/cerebro
chown -R cerebro:cerebro /opt/cerebro/
# create a directory for storing the log files
mkdir /var/log/cerebro

# change the hosts section in the config file /opt/cerebro/cerebro-0.6.4/conf/application.conf
hosts = [
  {
    host = "http://localhost:9200"
    name = "lure-box"
  }]
```

Listing A.4: Installation und Konfiguration des Elasticsearch-Webinterfaces Cerebro. [78]

A.2.3. Installation von Logstash

```
# update package repository und install Logstash
apt-get update && apt-get install logstash

## create a directory which is needed for last_run values
mkdir -p /etc/logstash/last_run
chown logstash:logstash /etc/logstash/last_run/
```

Listing A.5: Installation und Konfiguration von Logstash. [80]

A.2.4. Installation von Kibana

```
## installation of Kibana
# update package repository and install Kibana
apt-get update && apt-get install kibana
# create a directory for storing the log files anlegen
mkdir /var/log/kibana

## change the config file /etc/kibana/kibana.yml
# Kibana should listen on all available IP addresses
server.host: "0.0.0.0"
# change the name of the Kibana server. The name is only used for representation.
server.name: "Lure Box Kibana Server"
```

Listing A.6: Installation und Konfiguration von Kibana. [92]

A.2.5. ElastAlert

Installation

```
# install necessary dependencies
apt-get install virtualenv python-blist python-setuptools python-dev
 ↪  build-essential libssl-dev libffi-dev
# download the program
git clone https://github.com/Yelp/elastalert.git
# create a virtualenv environment in /opt
virtualenv /opt/elastalert
# activate virtualenv environment
source /opt/elastalert/bin/activate
# change into the elastalert directory
cd elastalert
# install all Python dependencies
(elastalert)root@elk-stack:~/elastalert pip install -r requirements.txt
(elastalert)root@elk-stack:~/elastalert# pip install requests==2.2.1
(elastalert)root@elk-stack:~/elastalert# pip install setuptools==34.4.1
# install ElastAlert
(elastalert)root@elk-stack:~/elastalert# python setup.py install
# create necessary index for Elasticsearch
(elastalert)root@elk-stack:~/elastalert# elastalert-create-index
Enter Elasticsearch host: 127.0.0.1
Enter Elasticsearch port: 9200
Use SSL? t/f: f
New index name? (Default elastalert_status)
Name of existing index to copy? (Default None)
New index elastalert_status created
Done!

# add a system user for ElastAlert
useradd -r elastalert
# create a directory for storing the config file
mkdir /etc/elastalert
# copy the example config file to /etc/elastalert
cp elastalert/config.yaml.example /etc/elastalert/config.yaml
# create a directory for storing rules
mkdir /etc/elastalert/rules
# set the correct file permissions
chown -R elastalert:elastalert /etc/elastalert/

# the following permission can be set after the file was created
chmod 400 smtp_auth_file.yaml
```

Listing A.7: Installation von ElastAlert in einer *virtualenv*-Umgebung. [163]

Konfiguration

```
# This is the folder that contains the rule yaml files
# Any .yaml file will be loaded as a rule
rules_folder: /etc/elastalert/rules
# How often ElastAlert will query Elasticsearch
# The unit can be anything from weeks to seconds
run_every:
  minutes: 1
# ElastAlert will buffer results from the most recent
# period of time, in case some log sources are not in real time
buffer_time:
  minutes: 15
# The Elasticsearch hostname for metadata writeback
# Note that every rule can have its own Elasticsearch host
es_host: 127.0.0.1
# The Elasticsearch port
es_port: 9200
# If an alert fails for some reason, ElastAlert will retry
# sending the alert until this time period has elapsed
alert_time_limit:
  days: 2
# Section for E-Mail alert
smtp_host: mail.fhstp.ac.at
smtp_port: 25
from_addr: sender@fhstp.ac.at
smtp_auth_file: '/etc/elastalert/smtp_auth_file.yaml'
```

Listing A.8: Inhalt der Konfigurationsdatei von ElastAlert.

Konfiguration der SMTP-Authentifizierung

```
user: elastalert@fhstp.ac.at
password: P@ssw0rd
```

Listing A.9: Inhalt der Datei für die SMTP-Authentifizierung von ElastAlert.

Testen einer Regel

```
# load virtualenv environment
source /opt/elastalert/bin/activate
# test a single rule
elastalert-test-rule /etc/elastalert/rules/monitor_mssql.yaml
# start ElastAlert with python
python -m elastalert.elastalert --verbose
```

Listing A.10: Testen einer einzelnen ElastAlert-Regel und das Starten von ElastAlert für alle Regeln.

A.2.6. Supervisor

Cerebro

```
[program:cerebro]
user=cerebro
command=/bin/bash -c "rm -f /opt/cerebro/cerebro-0.6.4/RUNNING_PID &&
  ↪  /opt/cerebro/cerebro-0.6.4/bin/cerebro -Dhttp.address=192.168.56.254"
process_name=cerebro
autostart=true
autorestart=true
startsecs=15
stopsignal=INT
stopasgroup=true
killasgroup=true
stderr_logfile=/var/log/cerebro/cerebro.stderr.log
stdout_logfile=/var/log/cerebro/cerebro.stdout.log
```

Listing A.11: Die Supervisor-Konfigurationsdatei für Cerebro.

Kibana

```
[program:kibana]
user=kibana
command=/usr/share/kibana/bin/kibana  -c /etc/kibana/kibana.yml
process_name=kibana
autostart=true
autorestart=true
startsecs=15
stopsignal=INT
stopasgroup=true
killasgroup=true
stderr_logfile=/var/log/kibana/kibana.stderr.log
stdout_logfile=/var/log/kibana/kibana.stdout.log
```

Listing A.12: Die Supervisor-Konfigurationsdatei für Kibana.

ElastAlert

```
[program:elastalert]
user=elastalert
command=/opt/elastalert/bin/elastalert --config /etc/elastalert/config.yaml
 ↪  --verbose
process_name=elastalert
autostart=true
autorestart=true
startsecs=15
stopsignal=INT
stopasgroup=true
killasgroup=true
stderr_logfile=/var/log/elastalert/elastalert.stderr.log
stdout_logfile=/var/log/elastalert/elastalert.stdout.log
```

Listing A.13: Die Supervisor-Konfigurationsdatei für ElastAlert.

A.3. Oracle-Datenbank

A.3.1. Chinook-Datenbank importieren

```
sqlplus sys/oracle as sysdba @Chinook_Oracle.sql > output
```

Listing A.14: Importieren der Chinook-Datenbank mittels *sqlplus*.

A.3.2. Salary-Tabelle anlegen

```
CREATE TABLE Salary
(
    EmployeeId NUMBER NOT NULL,
    Salary NUMBER NOT NULL,
    FromDate   DATE   NOT NULL,
    ToDate     DATE   NOT NULL,
    CONSTRAINT PK_Salary PRIMARY KEY  (EmployeeId)
);
ALTER TABLE Salary ADD CONSTRAINT FK_SalaryEmployee
    FOREIGN KEY (EmployeeId) REFERENCES Employee (Employeeid);

INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (1, 2000,
 ↪  TO_DATE('2015-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (2, 5000,
 ↪  TO_DATE('2016-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (3, 1500,
 ↪  TO_DATE('2016-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (4, 2000,
 ↪  TO_DATE('2015-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (5, 3000,
 ↪  TO_DATE('2014-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (6, 1000,
 ↪  TO_DATE('2016-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (7, 1000,
 ↪  TO_DATE('2016-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));
INSERT INTO Salary (EmployeeId, Salary, FromDate, ToDate) VALUES (8, 2000,
 ↪  TO_DATE('2016-12-01','YYYY-MM-DD'),TO_DATE('2099-12-12','YYYY-MM-DD'));

commit;
```

Listing A.15: Salary-Tabelle anlegen, auf die keine Person zugreifen darf, und ein paar Einträge hinzufügen.

A.3.3. Umschalten des Unified Auditing in den PURE-Modus

```
# 1. connect to the database via sqlplus. Use the user SYS with SYSDBA privileges
sqlplus sys as sysdba
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

# 2. shutdown database
SQL> shutdown immediate
Pluggable Database closed.
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
 ↪  Production

# 3. stop listener
lsnrctl stop
LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 05-MAR-2017 15:47:28
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
The command completed successfully

# 4. change into $ORACLE_HOME/rdbms/lib directory
racle@vbgeneric pfile]$ cd /u01/app/oracle/product/12.2/db_1/rdbms/lib

# 5. activate Unified Auditing
make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME

# 6. restart the whole server
```

Listing A.16: Konfigurieren des Unified Auditing in den PURE-Modus. [105]

```
SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified Auditing';
```

Listing A.17: Abfrage, die zeigt, ob Unified Auditing im PURE-Modus aktiv ist. [105]

A.3.4. Umschaltung der SGA in den Immediate-write-Modus

```
BEGIN
DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY(
  DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
  DBMS_AUDIT_MGMT.AUDIT_TRAIL_WRITE_MODE,
  DBMS_AUDIT_MGMT.AUDIT_TRAIL_IMMEDIATE_WRITE);
END;/ commit;
```

Listing A.18: Immediate-write-Modus für die SGA konfigurieren. [106]

A.3.5. Überwachung der Salary-Tabelle

```
-- deactivate and remove unified audit policy
NOAUDIT POLICY MONITOR_SALARY;
NOAUDIT POLICY MONITOR_SALARY by sys;
DROP AUDIT POLICY MONITOR_SALARY;
-- create policy
CREATE AUDIT POLICY MONITOR_SALARY ACTIONS
 ALTER on chinook.Salary,
 AUDIT on chinook.Salary,
 COMMENT on chinook.Salary,
 DELETE on chinook.Salary,
 FLASHBACK on chinook.Salary,
 GRANT on chinook.Salary,
 INDEX on chinook.Salary,
 INSERT on chinook.Salary,
 LOCK on chinook.Salary,
 RENAME on chinook.Salary,
 SELECT on chinook.Salary,
 UPDATE on chinook.Salary;
-- activate policy
AUDIT POLICY MONITOR_SALARY;
AUDIT POLICY MONITOR_SALARY by sys;
-- show information about the available unified audit policies
select distinct policy_name from audit_unified_policies;
select * from audit_unified_policies  where policy_name ='MONITOR_SALARY';
-- show information about all active unified audit policies
select * from audit_unified_enabled_policies;
-- save changes
commit;
```

Listing A.19: Konfiguration einer Unified Auditing Policy zur Überwachung jeglichen Zugriffs auf die Salary-Tabelle.

A.3.6. Überwachung von einzelnen Zeilen in der Employee-Tabelle

```
-- #### monitor the employee table with Fine Grained Auditing
--https://docs.oracle.com/database/121/DBSEG/audit_config.htm#DBSEG90057
-- deactivate policy
BEGIN
DBMS_FGA.DISABLE_POLICY(
  object_schema      => 'CHINOOK',
  object_name        => 'EMPLOYEE',
  policy_name        => 'MONITOR_EMPLOYEE');
END;
/
-- remove policy
BEGIN
 DBMS_FGA.DROP_POLICY(
  object_schema      => 'CHINOOK',
  object_name        => 'EMPLOYEE',
  policy_name        => 'MONITOR_EMPLOYEE');
END;
/

-- create policy
BEGIN
  DBMS_FGA.ADD_POLICY(
   object_schema      => 'CHINOOK',
   object_name        => 'EMPLOYEE',
   policy_name        => 'MONITOR_EMPLOYEE',
   audit_condition    => 'EMPLOYEEID in (4,5)',
   audit_column_opts   => DBMS_FGA.ANY_COLUMNS,
   enable             =>  TRUE,
   statement_types    => 'INSERT, UPDATE, SELECT, DELETE');
END;
/
-- save changes
commit;

-- show information about all available Fine Grained Audit Policies
SELECT policy_name, enabled, policy_text, policy_column_options from
 ↪  DBA_AUDIT_POLICIES;
```

Listing A.20: Konfiguration einer Fine Grained Auditing Policy zur Überwachung von Zugriffen auf Honeytokens in der Employee-Tabelle.

A.3.7. Installation eines Clients für den Remote-Zugriff

```
# Download client from
 ↪  http://www.oracle.com/technetwork/database/features/instant-client/

# Install necessary library
apt-get install libaio1
echo "/usr/lib/oracle/12.2/client64/lib/" > /etc/ld.so.conf.d/oracle.conf

# Convert and Install downloaded packages
alien -i oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm
alien -i oracle-instantclient12.2-devel-12.2.0.1.0-1.x86_64.rpm
alien -i oracle-instantclient12.2-sqlplus-12.2.0.1.0-1.x86_64.rpm
```

Listing A.21: Herunterladen und Installation des Clients *sqlplus*. [164]

A.3.8. Remote-Zugriff

```
sqlplus64 chinook/p4ssw0rd@//192.168.56.1:1521/orcl

SQL> select * from Salary;

EMPLOYEEID     SALARY FROMDATE  TODATE
---------- ---------- --------- ---------
         1       2000 01-MAR-15 12-DEC-99
         2       5000 01-DEC-16 12-DEC-99
         3       1500 01-DEC-16 12-DEC-99
...
SQL> select * from Employee;
...
```

Listing A.22: Zugriff von der Lure Box auf die Honeytokens in der Oracle-Datenbank.

A.3.9. Logs für Salary-Tabelle ansehen

```
-- show the logs about users accessing the Salary table
 select event_timestamp, action_name, system_privilege_used, object_name
  from unified_audit_trail
  where object_name = 'MONITOR_SALARY'
  or object_name = 'SALARY'
  order by event_timestamp;

select event_timestamp, os_username, userhost, terminal,
 ↪  dbusername,client_program_name, action_name, sql_text, object_name
  from unified_audit_trail
  where unified_audit_policies='MONITOR_SALARY'
  order by event_timestamp;

select event_timestamp, dbusername,object_name,action_name from unified_audit_trail
  where unified_audit_policies='MONITOR_SALARY' and event_timestamp >
   ↪  trunc(sysdate)
  order by event_timestamp;
```

Listing A.23: Ansehen der Logs, die aufgrund von Zugriffen auf die Salary-Tabelle generiert wurden.

A.3.10. Logs für die Employee-Tabelle ansehen

```
-- show the logs about users accessing the Employee table
select event_timestamp, os_username, userhost, terminal,
 ↪  dbusername,client_program_name, action_name, sql_text, object_name
  from unified_audit_trail
  where fga_policy_name='MONITOR_EMPLOYEE'
  order by event_timestamp desc;
```

Listing A.24: Ansehen der Logs, die aufgrund von Zugriffen auf die Employee-Tabelle generiert wurden.

A.3.11. Anbindung an die Lure Box

Konto für den Remote-Zugriff anlegen

```
-- create a user called auditor with the password oracle
CREATE USER auditor IDENTIFIED BY oracle;

-- assign the role audit_viewer to the user
 grant create session, audit_viewer to auditor;

commit;

-- test the connection from the Lure Box to the database with the newly created
 ↪  user
sqlplus64 auditor/oracle@//192.168.56.1:1521/orcl
SQL> select * from unified_audit_trail;
...
```

Listing A.25: Anlegen und Testen eines Kontos mit der Berechtigung für den Zugriff auf Logdaten.

Installation des JDBC-Treibers

```
# download the JDBC driver from the Oracle website
http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html

# create a directory to store the driver
mkdir -p /opt/jdbc/oracle
# move the driver into the directory /opt/jdbc/oracle
mv ojdbc8.jar /opt/jdbc/oracle
```

Listing A.26: Herunterladen und Abspeichern des Oracle JDBC-Treibers. [109]

Logstash

```
input {
        # use jdbc as input for this log source
        jdbc {
                jdbc_validate_connection => true
                jdbc_connection_string =>
                 ↪  "jdbc:oracle:thin:@192.168.56.1:1521/orcl"
                jdbc_user => "auditor"
                jdbc_password => "oracle"
```

```
8                 jdbc_driver_library => "/opt/jdbc/oracle/ojdbc8.jar"
9                 jdbc_driver_class => "Java::oracle.jdbc.driver.OracleDriver"
10                statement => "select event_timestamp, audit_type,
                   ↪  unified_audit_policies, fga_policy_name, action_name,
                   ↪  return_code, object_schema, object_name, sql_text,
11                os_username, userhost, client_program_name, terminal, dbid,
    ↪  authentication_type, dbusername, entry_id from unified_audit_trail where
    ↪  event_timestamp >= :sql_last_value"
12                clean_run => "false"
13                record_last_run => "true"
14                last_run_metadata_path =>
                   ↪  "/etc/logstash/last_run/jdbc_last_run_oracle"
15                schedule => "* * * * *"
16        }
17 }
18 filter {
19         grok {
20                match => { "authentication_type" => "\TYPE=\(%{WORD}\)\);\(CLIENT
                   ↪  ADDRESS=\(\(ADDRESS=\(PROTOCOL=%{WORD:protocol}\)
                   ↪  \(HOST=%{IP:srcip}\)\(PORT=%{INT:srcport}\)\)\)\)\);" }
21         }
22
23         # add information about the origin of the log source
24         mutate { add_field => ["[log_source][received_at]", "%{@timestamp}"] }
25         mutate { add_field => ["[log_source][host]", "192.168.56.1"] }
26         mutate { add_field => ["[log_source][product]", "Oracle Database 12c"] }
27
28         # summarize fields about the source
29         mutate { rename => {"[userhost]" => "[src][hostname]"} }
30         mutate { rename => {"[terminal]" => "[src][terminal]"} }
31         mutate { rename => {"[srcip]" => "[src][ip]"} }
32         mutate { rename => {"[os_username]" => "[src][client_username]"} }
33         mutate { rename => {"[srcport]" => "[src][port]"} }
34         mutate { rename => {"[protocol]" => "[src][protocol]"} }
35         mutate { rename => {"[client_program_name]" => "[src][client_program]"} }
36
37         # summarize fields about the audit
38         mutate { rename => {"[action_name]" => "[audit][action]"} }
39         mutate { rename => {"[audit_type]" => "[audit][type]"} }
40         mutate { rename => {"[object_name]" => "[audit][object][name]"} }
41         mutate { rename => {"[dbusername]" => "[audit][database][username]"} }
42         mutate { rename => {"[dbid]" => "[audit][database][id]"} }
43         mutate { rename => {"[object_schema]" => "[audit][object][schema]"} }
44         mutate { rename => {"[sql_text]" => "[audit][query][text]"} }
45         mutate { rename => {"[return_code]" => "[audit][query][return_code]"} }
46         mutate { rename => {"[unified_audit_policies]" => "[audit][policy]"} }
47         if [fga_policy_name] { # check if the field fga_policy_name is existing
48                 mutate { rename => {"[fga_policy_name]" => "[audit][policy]"} }
49         }
50
```

```
51          mutate { rename => {"[entry_id]" => "[ID]"} }
52
53          # use the field event_timestamp as timestamp for the event
54          mutate { convert => ["event_timestamp" , "string"] }
55          date { match => ["event_timestamp", "ISO8601"] }
56
57           # remove unnecessary fields
58          mutate { remove_field =>
             ↪  ["event_timestamp","authentication_type","fga_policy_name","ID"] }
59  }
60  output {
61          # uncomment the following line for getting a debug output in logstash
62          # stdout { codec => rubydebug}
63
64          # write the output to the elasticsearch datbase
65          elasticsearch {
66                  hosts => ["127.0.0.1:9200"]
67                  index => "audit_databases_oracle-%{+YYYY.MM.dd}"
68          }
69  }
```

Listing A.27: Die Logstash-Konfigurationsdatei */etc/logstash/conf.d/oracle_database.conf*.

```
1  mkdir -p /usr/share/logstash/data/oracle
2  chown logstash:logstash /usr/share/logstash/data/oracle
```

Listing A.28: Ein Datenverzeichnis für die neue Logstash-Instanz anlegen.

```
1  [program:logstash_oracle]
2  directory=/
3  user=logstash
4  command=/usr/share/logstash/bin/logstash -f
    ↪  /etc/logstash/conf.d/oracle_database.conf --path.data
    ↪  /usr/share/logstash/data/oracle/
5  autostart=true
6  autorestart=true
7  stderr_logfile=/var/log/logstash/logstash_oracle.stderr.log
8  stdout_logfile=/var/log/logstash/logstash_oracle.stdout.log
```

Listing A.29: Die Supervisor-Konfigurationsdatei für die Abholung von Oracle-Logs mittels Logstash.

ElastAlert

```
# Rule name, must be unique
name: Monitor Oracle

# Type of alert.
# the frequency rule type alerts when num_events events occur with timeframe time
type: any

# Index to search, wildcard supported
index: audit_databases_oracle-*

filter:
- query:
    query_string:
      query: "audit.policy:MONITOR_SALARY"

query_key: "[audit][policy]"
realert:
 minutes: 10
# The alert is use when a match is found
alert:
- "email"

alert_subject: "ElastAlert: Access to monitored object {1} detected {0}."
alert_subject_args:
- "@timestamp"
- audit.object.name

alert_text_args:
- "@timestamp"
- audit.object.name

alert_text: |
 Access to to the monitored object {1} detected.
 Connect to the Kibana webinterface http://192.168.56.254:5601 to receive more
  ↪ information.
 You can also find further information about the event below.
# a list of email addresses to send alerts to
email:
- "incident@fhstp.ac.at"
```

Listing A.30: Die ElastAlert-Regel */etc/elastalert/rules/monitor_oracle.yaml* zur Alarmierung von Zugriffen auf Honeytokens.

A.4. Microsoft SQL Server

A.4.1. Chinook-Datenbank importieren

```
sqlcmd -i Chinook_SqlServer_AutoIncrementPKs.sql -o output.txt
```

Listing A.31: Importieren der Chinook-Datenbank mittels *sqlcmd*. [113]

A.4.2. Salary-Tabelle anlegen

```
CREATE TABLE [dbo].[Salary]
(
    [EmployeeId] INT NOT NULL,
    [Salary] INT NOT NULL,
    [FromDate] DATE NOT NULL,
    [ToDate] DATE NOT NULL,
    CONSTRAINT [PK_Salary] PRIMARY KEY CLUSTERED ([EmployeeId])
);
/* create foreign key */
ALTER TABLE [dbo].[Salary] ADD CONSTRAINT [FK_SalaryEmployee]
    FOREIGN KEY ([EmployeeId]) REFERENCES [dbo].[Employee] ([EmployeeId]);

INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (1,2000,'2015-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (2,5000,'2016-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (3,1500,'2016-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (4,2000,'2015-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (5,3000,'2014-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (6,1000,'2016-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (7,1000,'2016-12-01','2099-12-12');
INSERT INTO [dbo].[Salary] ([EmployeeId], [Salary], [FromDate], [ToDate]) VALUES
 ↪  (8,2000,'2016-12-01','2099-12-12');
```

Listing A.32: Salary-Tabelle anlegen, auf die keine Person zugreifen darf, und ein paar Einträge hinzufügen.

A.4.3. Überwachung der Salary-Tabelle

```
-- switch to the master context
USE master;
GO
-- create a server audit object
CREATE SERVER AUDIT MyServerAudit
TO FILE
( FILEPATH = N'C:\fhstp\Audit'
 ,MAXSIZE = 100 MB
 ,MAX_ROLLOVER_FILES = 2147483647
 ,RESERVE_DISK_SPACE = OFF
)
WITH
( QUEUE_DELAY = 1000
 ,ON_FAILURE = CONTINUE
)
GO
```

Listing A.33: Anlegen eines Server-Audit-Objektes.

```
-- switch to the Chinook context
USE Chinook;

CREATE DATABASE AUDIT SPECIFICATION Monitor_Salary_Employee
FOR SERVER AUDIT MyServerAudit
ADD (SELECT,INSERT, UPDATE, DELETE, EXECUTE, RECEIVE ON dbo.Salary by public),
ADD (SELECT,INSERT, UPDATE, DELETE, EXECUTE, RECEIVE ON dbo.Employee by public)
WITH (STATE = ON);
GO
```

Listing A.34: Anlegen der Datenbank-Audit-Spezifikation zur Überwachung der Salary- und der Employee-Tabelle.

```
USE master;
GO
Alter Server Audit MyServerAudit with(State=ON) -- deaktivieren mittels STATE=OFF
GO
```

Listing A.35: Aktivieren des Server-Audit-Objektes *MyServerAudit*.

A.4.4. Installation eines Clients für den Remote-Zugriff

```
# installation of the necessary packages
apt-get install freetds-common freetds-bin unixodbc

# definition of the SQL server in the FreeTDS config file /etc/freetds/freetds.conf
# A typical Microsoft server
[MSSQLServer]
        host = 192.168.56.2
        port = 1433
        tds version = 8.0
```

Listing A.36: Installation und Konfiguration von FreeTDS für den Zugriff auf eine MSSQL-Datenbank.

A.4.5. Remote-Zugriff

```
# -S name of the server which is defined in/etc/freetds/freetds.conf
# -U username for the connection
# -P password for the connection
# -D database to connect to
tsql -S MSSQLServer -U auditor -P P@ssw0rd -D Chinook

1> select * from Salary;
2> go
EmployeeId      Salary  FromDate        ToDate
1       2000    2015-12-01      2099-12-12
...
1> select * from Employee;
2> go
...
```

Listing A.37: Zugriff von der Lure Box auf die Honeytokens in der MSSQL-Datenbank.

A.4.6. Logs für Salary- und Employee-Tabelle ansehen

```
SELECT event_time,server_principal_name,database_principal_name, object_name,
 ↪  statement,* FROM fn_get_audit_file ('C:\fhstp\Audit\*',default, default)
```

Listing A.38: Ansehen der Logs, die aufgrund von Zugriffen auf die Salary- und die Employee-Tabelle generiert wurden.

A.4.7. Anbindung an die Lure Box

Konto für den Remote-Zugriff anlegen

```
-- create a user called auditor with the password P@ssw0rd
create login auditor WITH PASSWORD = 'P@ssw0rd',
DEFAULT_DATABASE = Chinook;
GO
use master;
grant CONTROL SERVER to auditor;
GO

-- test the connection from the Lure Box to the database with the newly created
 ↪  user
tsql -S MSSQLServer -U auditor -P P@ssw0rd -D Chinook
SELECT event_time,server_principal_name,database_principal_name, object_name,
 ↪  statement,* FROM fn_get_audit_file ('C:\fhstp\Audit\*',default, default)
1> SELECT event_time,server_principal_name,database_principal_name, object_name,
 ↪  statement,* FROM fn_get_audit_file ('C:\fhstp\Audit\*',default, default)
2> GO
...
```

Listing A.39: Anlegen und Testen eines Kontos mit der Berechtigung für den Zugriff auf Logdaten.

Installation des JDBC-Treibers

```
# create a directory to store the driver
mkdir -p /opt/jdbc/mssql
# extact the driver and move it into the directory /opt/jdbc/mssql
tar xvf sqljdbc_6.0.8112.100_enu.tar.gz
mv sqljdbc_6.0/enu/jre/sqljdbc42.jar /opt/jdbc/mssql/
```

Listing A.40: Herunterladen und Abspeichern des MSSQL JDBC-Treibers. [123]

Logstash

```
input {
        # use jdbs as input for this log source
        jdbc {
                jdbc_validate_connection => true
                jdbc_connection_string =>
                 ↪  "jdbc:sqlserver://192.168.56.2:1433;databaseName=Chinook"
```

```
6                   jdbc_user => "auditor"
7                   jdbc_password => "P@ssw0rd"
8                   jdbc_driver_library => "/opt/jdbc/mssql/sqljdbc42.jar"
9                   jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
10                  statement => "select dateadd(hour,2,cast(event_time as datetime))
                     ↪  as event_time, action_id, session_id, server_principal_id,
                     ↪  object_id, server_principal_name,
                     ↪  database_principal_name,server_instance_name, database_name,
                     ↪  schema_name, object_name, statement,  succeeded, file_name,
                     ↪  audit_file_offset from fn_get_audit_file
                     ↪  ('C:\fhstp\Audit\*',default, default);"
11                  clean_run => "false"
12                  record_last_run => "true"
13                  last_run_metadata_path =>
                     ↪  "/etc/logstash/last_run/jdbc_last_run_mssql"
14                  schedule => "* * * * *"
15          }
16  }
17  filter {
18           # add information about the origin of the log source
19          mutate { add_field => ["[log_source][received_at]", "%{@timestamp}"] }
20          mutate { add_field => ["[log_source][host]", "192.168.56.2"] }
21          mutate { add_field => ["[log_source][product]", "Microsoft SQL Server
             ↪  2016"] }
22
23          # summarize fields about the audit
24          mutate { rename => {"[session_id]" => "[audit][session_id]"} }
25          mutate { rename => {"[action_id]" => "[audit][action]"} }
26          mutate { rename => {"[server_instance_name]" =>
             ↪  "[audit][server][instance_name]"} }
27          mutate { rename => {"[server_principal_id]" =>
             ↪  "[audit][database][user_id]"} }
28          mutate { rename => {"[server_principal_name]" =>
             ↪  "[audit][database][username]"} }
29          mutate { rename => {"[database_principal_name]" =>
             ↪  "[audit][database][principal_name]"} }
30          mutate { rename => {"[database_name]" => "[audit][database][name]"} }
31          mutate { rename => {"[file_name]" => "[audit][file][name]"} }
32          mutate { rename => {"[audit_file_offset]" => "[audit][file][offset]"} }
33          mutate { rename => {"[object_name]" => "[audit][object][name]"} }
34          mutate { rename => {"[object_id]" => "[audit][object][id]"} }
35          mutate { rename => {"[schema_name]" => "[audit][object][schema]"} }
36          mutate { rename => {"[statement]" => "[audit][query][text]"} }
37          mutate { rename => {"[succeeded]" => "[audit][query][succeeded]"} }
38
39          # use the event_time field as timestamp for the event
40          # attention: in the SQL query two hours were added on purpose.
41          # this was necessary to get the correct timestamp stored in the database
42          mutate { convert => ["event_time" , "string"] }
43          date { match => ["event_time", "ISO8601"]  }
```

```
44
45          # remove unnecessary fields
46          mutate { remove_field => ["event_time"] }
47
48  }
49  output {
50          # uncomment the following line for getting a debug output in logstash
51          #stdout { codec => rubydebug}
52
53          # write the output to the elasticsearch database
54          elasticsearch {
55                  hosts => ["127.0.0.1:9200"]
56   index => "audit_databases_mssql-%{+YYYY.MM.dd}"
57          }
58  }
```

Listing A.41: Die Logstash-Konfigurationsdatei */etc/logstash/conf.d/mssql_database.conf*

```
1  mkdir -p /usr/share/logstash/data/mssql
2  chown logstash:logstash /usr/share/logstash/data/mssql
```

Listing A.42: Ein Datenverzeichnis für die neue Logstash-Instanz anlegen.

```
1  [program:logstash_mssql]
2  directory=/
3  user=logstash
4  command=/usr/share/logstash/bin/logstash -f
    ↪  /etc/logstash/conf.d/mssql_database.conf --path.data
    ↪  /usr/share/logstash/data/mssql/
5  autostart=true
6  autorestart=true
7  stderr_logfile=/var/log/logstash/logstash_mssql.stderr.log
8  stdout_logfile=/var/log/logstash/logstash_mssql.stdout.log
```

Listing A.43: Die Supervisor-Konfigurationsdatei für die Abholung von Microsoft SQL-Server-Logs mittels Logstash.

ElastAlert

```
# Rule name, must be unique
name: Monitor MSSQL

# Type of alert.
# the frequency rule type alerts when num_events events occur with timeframe time
type: any

# Index to search, wildcard supported
index: audit_databases_mssql-*

filter:
- query:
    query_string:
      query: "audit.object.name:Salary"

query_key: "[audit][object][name]"

realert:
 minutes: 10
# (Required)
# The alert is use when a match is found
alert:
- "email"

alert_subject: "ElastAlert: Access to monitored object {1} detected {0}."
alert_subject_args:
- "@timestamp"
- audit.object.name

alert_text_args:
- "@timestamp"
- audit.object.name

alert_text: |
 Access to to the monitored object {1} detected.
 Connect to the Kibana webinterface http://192.168.56.254:5601 to receive more
  ↪ information.
 You can also find further information about the event below.
# (required, email specific)
# a list of email addresses to send alerts to
email:
- "incident@fhstp.ac.at"
```

Listing A.44: Die ElastAlert-Regel */etc/elastalert/rules/monitor_mssql.yaml* zur Alarmierung von Zugriffen auf Honeytokens.

A.5. PostgreSQL

A.5.1. Chinook-Datenbank importieren

```
# create a user called chinookuser
# -P tells postgres to ask for a password for the user
# -d gives the user the right to create new databases
sudo -u postgres createuser -P -d chinookuser

# create database Chinook with the owner chinookuser
sudo -u postgres createdb -O chinookuser Chinook

# import the database out of the file Chinook_PostgreSql.sql
sudo -u postgres psql -f Chinook_PostgreSql.sql -q Chinook
```

Listing A.45: Importieren der Chinook-Datenbank mittels *psql*.

A.5.2. Salary-Tabelle anlegen

```
# connect to the postgres database Chinook via psql
sudo -u postgres psql -d Chinook

CREATE TABLE "Salary"
(
        "EmployeeId" INT NOT NULL,
        "Salary" INT NOT NULL,
        "FromDate" TIMESTAMP NOT NULL,
        "ToDate" TIMESTAMP NOT NULL,
        CONSTRAINT "PK_Salary" PRIMARY KEY  ("EmployeeId")
);

ALTER TABLE "Salary" ADD CONSTRAINT "FK_SalaryEmployee"
        FOREIGN KEY ("EmployeeId") REFERENCES "Employee" ("EmployeeId") ON DELETE
          ↪  NO ACTION ON UPDATE NO ACTION;

INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (1,
 ↪  2000,'2015/12/01', '2099-12-12');
INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (2,
 ↪  5000,'2015/12/01', '2099-12-12');
INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (3,
 ↪  1500,'2015/12/01', '2099-12-12');
INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (4,
 ↪  2000,'2015/12/01', '2099-12-12');
INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (5,
 ↪  3000,'2015/12/01', '2099-12-12');
INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (6,
 ↪  1000,'2015/12/01', '2099-12-12');
```

```
22 INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (7,
    ↪  1000,'2015/12/01', '2099-12-12');
23 INSERT INTO "Salary" ("EmployeeId", "Salary", "FromDate", "ToDate") VALUES (8,
    ↪  2000,'2015/12/01', '2099-12-12');
```

Listing A.46: Salary-Tabelle anlegen, auf die keine Person zugreifen darf, und ein paar Einträge hinzufügen.

A.5.3. Überwachung aller Tabellen

```
1  # ERROR REPORTING AND LOGGING
2  log_destination = 'csvlog'
3  logging_collector = on
4
5  # the path for the log files can be defined absolute or relative to the PGDATA
    ↪  directory
6  log_directory = '/var/log/postgresql/pg_log/'
7  log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # pattern for name of the log files
8
9  log_file_mode = 0640
10
11 log_truncate_on_rotation = off
12 log_rotation_age = 1d                    # automatic rotation of the log file after
    ↪  1 day
13 log_rotation_size = 10MB                 # automatic rotation of the log file when
    ↪  the file reaches a size bigger than 10 MB
14
15 # - how should be logged-
16 log_line_prefix = '%t [%p-%l] %r %q%u@%d '
17
18 # explanation of the prefixes:
19 # %t = timestamp in ms
20 # %p = process ID
21 # %l = line number of the session
22 # %r = remote host and port
23 # %u = username
24 # %d = database name
25
26 #log_lock_waits = off
27 log_statement = 'all'
28 log_timezone = 'localtime'
```

Listing A.47: Ausschnitt aus der Konfiguratoinsdatei von PostgreSQL. In diesem Abschnitt wird das Auditing konfiguriert.

```
service postgresql restart
```

Listing A.48: Neustart des Datenbankservers durchführen, damit Änderungen aktiv werden.

A.5.4. Vorbereitung von PostgreSQL für den Remote-Zugriff

```
# switch to user postgres
su - postgres

# connect to the chinook database via psql
psql Chinook

# assign necessary privileges to chinookuser
Chinook=# grant select on all tables in schema public to chinookuser;
Chinook=# grant connect on database "Chinook" to chinookuser;
Chinook=# grant select on all sequences in schema public to chinookuser;
```

Listing A.49: Vergabe der für den Zugriff erforderlichen Berechtigungen an das Konto *chinookuser*.

```
# change network listen address in /etc/postgresql/9.5/main/postgresql.conf
listen_addresses = 'localhost,192.168.56.3'

# change authentication in /etc/postgresql/9.5/main/pg_hba.conf
# the entries have the format type,database, user, address and method

# all local connections shall be authenticated via md5
local   all             all                                     md5

# IPv4 connections from the local computer are fully trusted
# the host 192.168.56.254 is allowed to connect to the Chinook database via md5
host    all             all             127.0.0.1/32            trust
host Chinook chinookuser 192.168.56.254/32 md5
```

Listing A.50: Konfiguration von PostgreSQL für den Remote-Zugriff von der Lure Box.

A.5.5. Installation eines Clients für den Remote-Zugriff

```
apt-get install postgresql-client
```

Listing A.51: Installation eines PostgreSQL-Clients auf der Lure Box.

A.5.6. Remote-Zugriff

```
# connect to database and query salary table
psql -h 192.168.56.3 Chinook chinookuser

Chinook=> select * from "Salary";
 EmployeeId | Salary |      FromDate       |       ToDate
------------+--------+---------------------+---------------------
          1 |   2000 | 2015-12-01 00:00:00 | 2099-12-12 00:00:00
          2 |   5000 | 2015-12-01 00:00:00 | 2099-12-12 00:00:00
```

Listing A.52: Zugriff von der Lure Box auf die Honeytokens in der PostgreSQL-Datenbank.

A.5.7. Anbindung an die Lure Box

Konfiguration von Filebeat

```
#========================== Filebeat prospectors =============================
filebeat.prospectors:
- input_type: log
  # paths where to look for log files
  paths:
    - /var/log/postgresql/pg_log/*.csv

 ### Multiline options
  # the regex pattern is used to detected new lines
  multiline.pattern: '^[0-9]{4}-[0-9]{2}-[0-9]{2}'
  multiline.negate: true
  multiline.match: after
#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.56.254:5044"]
```

Listing A.53: Änderungen, die an der Konfigurationsdatei von Filebeat vorgenommen werden müssen.

```
# add user filebeat to the postgres group
usermod -a -G postgres filebeat

# change permissions of the log directory
chmod 751 /var/log/postgresql/pg_log/
```

Listing A.54: Filebeat als Mitglied der Gruppe *postgres* definieren und die Berechtigungen des PostgreSQL-Logverzeichnisses anpassen.

```
[program:filebeat_postgresql]
directory=/
user=filebeat
command=/usr/share/filebeat/bin/filebeat -e
 -c /etc/filebeat/filebeat.yml
 -path.home /usr/share/filebeat
 -path.config /etc/filebeat
 -path.data /var/lib/filebeat
 -path.logs /var/log/filebeat
autostart=true
autorestart=true
stderr_logfile=/var/log/filebeat/filebeat_postgresql.stderr.log
stdout_logfile=/var/log/filebeat/filebeat_postgresql.stdout.log
```

Listing A.55: Die Supervisor-Konfigurationsdatei für das Versenden der PostgreSQL-Logs mittels Filebeat an die Lure Box.

Logstash

```
input {
        # use filebeat as input for this log source
        beats {
                port => 5044
                ssl => false
        }
}
filter {
        # add information about the origin of the log source
        mutate { add_field => ["[log_source][received_at]", "%{@timestamp}"] }
        mutate { add_field => ["[log_source][host]", "192.168.56.3"] }
        mutate { add_field => ["[log_source][product]", "PostgreSQL 9.5.6"] }

        # split the CSV message in it's various fields
```

```
15          csv {
16                  columns => ["log_time","user_name","database_name","process_id",
                     ↪  "connection_from","session_id","session_line_num",
                     ↪  "command_tag","session_start_time","virtual_transaction_id",
                     ↪  "transaction_id","error_severity","sql_state_code","message",
                     ↪  "detail","hint","internal_query","internal_query_pos",
                     ↪  "context","query","query_pos","location","application_name"]
17                  separator => ","
18          }
19
20          # split up source IP und source port
21          grok {
22                  match => { "connection_from" => "%{IP:srcip}:%{INT:srcport}" }
23          }
24          # parse the SQL statements out of the message field
25          if ([message] =~ /statement:.*/) {
26                  grok {
27                          match => { "message" => "statement: %{GREEDYDATA:sql_text}"
                             ↪  }
28                  }
29                  mutate { rename => {"[sql_text]" => "[audit][query][text]"} }
30          }
31
32          # summarize fields about the source
33          mutate { rename => {"[srcip]" => "[src][ip]"} }
34          mutate { rename => {"[srcport]" => "[src][port]"} }
35          mutate { rename => {"[application_name]" => "[src][client_program]"} }
36
37          # summarize fields about the audit
38          mutate { rename => {"[session_id]" => "[audit][session_id]"} }
39          mutate { rename => {"[context]" => "[audit][context]"} }
40          mutate { rename => {"[detail]" => "[audit][detail]"} }
41          mutate { rename => {"[error_severity]" => "[audit][error_severity]"} }
42          mutate { rename => {"[message]" => "[audit][message]"} }
43          mutate { rename => {"[host]" => "[audit][server][instance_name]"} }
44          mutate { rename => {"[database_name]" => "[audit][database][name]"} }
45          mutate { rename => {"[user_name]" => "[audit][database][username]"} }
46          mutate { rename => {"[location]" => "[audit][query][location]"} }
47          mutate { rename => {"[sql_state_code]" => "[audit][query][state_code]"} }
48          mutate { rename => {"[internal_query]" => "[audit][query][internal_query]"}
             ↪  }
49          mutate { rename => {"[source]" => "[audit][file][name]"} }
50          mutate { rename => {"[offset]" => "[audit][file][offset]"} }
51
52
53          # use the field session_start_time as timestamp for the event
54          mutate { convert => ["session_start_time" , "string"] }
55          date { match => ["session_start_time", "YYYY-MM-dd HH:mm:ss 'CEST'"] }
56
57          # remove unnecessary fields
```

```
58          mutate { remove_field =>
              ↪  ["[beat][name]","[beat][version]","[beat][hostname]","hint",
              ↪  "virtual_transaction_id","transaction_id","query_pos","query",
              ↪  "process_id","internal_query_pos","input_type","type",
              ↪  "session_line_num","command_tag","log_time","session_start_time",
              ↪  "connection_from"] }
59  }
60  output {
61          # uncomment the following line for getting a debug output in logstash
62          #stdout { codec => rubydebug}
63
64          # write the output to the elasticsearch database
65          elasticsearch {
66                  hosts => ["127.0.0.1:9200"]
67                  index => "audit_databases_postgresql-%{+YYYY.MM.dd}"
68          }
69  }
```

Listing A.56: Die Logstash-Konfigurationsdatei */etc/logstash/conf.d/postgresql_database.conf*

```
1  mkdir -p /usr/share/logstash/data/postgresql
2  chown logstash:logstash /usr/share/logstash/data/postgresql
```

Listing A.57: Ein Datenverzeichnis für die neue Logstash-Instanz anlegen.

```
1  [program:logstash_postgresql]
2  directory=/
3  user=logstash
4  command=/usr/share/logstash/bin/logstash -f
    ↪  /etc/logstash/conf.d/postgresql_database.conf --path.data
    ↪  /usr/share/logstash/data/postgresql/
5  autostart=true
6  autorestart=true
7  stderr_logfile=/var/log/logstash/logstash_postgresql.stderr.log
8  stdout_logfile=/var/log/logstash/logstash_postgresql.stdout.log
```

Listing A.58: Die Supervisor-Konfigurationsdatei für die Verarbeitung der PostgreSQL-Logs mittels Logstash.

ElastAlert

```
# Rule name, must be unique
name: Monitor PostgreSQL

# Type of alert.
# the frequency rule type alerts when num_events events occur with timeframe time
type: any

# Index to search, wildcard supported
index: audit_databases_postgresql-*

filter:
- query:
    query_string:
      query: "audit.query.text:Salary"

query_key: "[audit][message]"

realert:
 minutes: 10
# (Required)
# The alert is use when a match is found
alert:
- "email"

alert_subject: "ElastAlert: Access to monitored object {1} detected {0}"
alert_subject_args:
- "@timestamp"
- audit.database.name

alert_text_args:
- "@timestamp"

alert_text: |
 Access to to the monitored object detected.
 Connect to the Kibana webinterface http://192.168.56.254:5601 to receive more
  ↪  information.
 You can also find further information about the event below.
# a list of email addresses to send alerts to
email:
- "incident@fhstp.ac.at"
```

Listing A.59: Die ElastAlert-Regel */etc/elastalert/rules/monitor_mssql.yaml* zur Alarmierung von Zugriffen auf Honeytokens.

A.6. MongoDB

A.6.1. Importieren einer Datenbank

```
# download the database
wget https://raw.githubusercontent.com/mongodb/
  ↪  docs-assets/primer-dataset/primer-dataset.json

# import the database
mongoimport --db fhstp --collection restaurants --drop --file primer-dataset.json
```

Listing A.60: Importieren einer Beispieldatenbank für MongoDB. [139]

A.6.2. Salary-Tabelle anlegen

```
# start the MongoDB shell
mongo
# switch to the fhstp database
use fhstp
# create Salary collection
db.createCollection("Salary")
# insert data into the collection
db.salary.insert(
    [
      { EmployeeId: 1, Salary: 2000, FromDate: ISODate("2015-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 2, Salary: 5000, FromDate: ISODate("2016-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 3, Salary: 1500, FromDate: ISODate("2016-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 4, Salary: 2000, FromDate: ISODate("2015-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 5, Salary: 3000, FromDate: ISODate("2014-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 6, Salary: 1000, FromDate: ISODate("2016-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 7, Salary: 1000, FromDate: ISODate("2016-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") },
      { EmployeeId: 8, Salary: 2000, FromDate: ISODate("2016-12-01T00:00:00Z"),
       ↪  ToDate: ISODate("2099-12-12T00:00:00Z") }
    ]
)
```

Listing A.61: Anlegen einer Salary-Collection und das Einfügen von Datensätzen in die Collection.

A.6.3. Überwachung der Salary- und Restaurants-Collections

```
## Enterprise-Only Options:
auditLog:
  destination: file
  format: JSON
  path: /var/lib/mongodb/auditLog.json
  filter: '{ atype: "authCheck", "param.ns": { $in: ["fhstp.salary",
    ↪  "fhstp.restaurants"]}, "param.command": { $in: [ "find", "insert", "delete",
    ↪  "update", "findandmodify" ] } }'
setParameter: { auditAuthorizationSuccess: true }
```

Listing A.62: Ausschnitt aus der Konfigurationsdatei von MongoDB. In der *auditLog*-Sektion wird dem Daemon mitgeteilt, dass alle auf die Collections „Salary“ und „Restaurants“ abzielenden Zugriffe auditiert werden sollen.

A.6.4. Vorbereitung von MongoDB für den Remote-Zugriff

```
mongo
MongoDB Enterprise > use fhstp
MongoDB Enterprise >    db.createUser({
                        user: 'chinookuser',
                        pwd: 'P@ssw0rd',
                        roles: [{ role: 'readWrite', db:'fhstp'}]
                        })
exit
```

Listing A.63: Konto für den entfernten Zugriff auf MongoDB anlegen.

```
# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1,192.168.56.4

security:
  authorization: 'enabled'
```

Listing A.64: MongoDB aus dem Netzwerk 192.168.56.0/24 erreichbar machen und den BenutzerInnen die Authentifizierung ermöglichen.

A.6.5. Installation eines Clients für den Remote-Zugriff

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
  ↪  0C49F3730359A14518585931BC711F9BA15703C6
echo "deb http://repo.mongodb.com/apt/debian jessie/mongodb-enterprise/3.4 main" |
  ↪  sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list
sudo apt-get update
sudo apt-get install mongodb-enterprise-shell
```

Listing A.65: Installation der MongoDB-Shell über die Paketquellen. [165]

A.6.6. Remote-Zugriff

```
mongo 192.168.56.4/fhstp -u chinookuser -p P@ssw0rd
MongoDB Enterprise > db.salary.find()
{ "_id" : ObjectId("58da897c29b4c2af9c4de080"), "EmployeeId" : 1, "Salary" : 2000,
  ↪  "FromDate" : ISODate("2015-12-01T00:00:00Z"), "ToDate" :
  ↪  ISODate("2099-12-12T00:00:00Z") }
{ "_id" : ObjectId("58da897c29b4c2af9c4de081"), "EmployeeId" : 2, "Salary" : 5000,
  ↪  "FromDate" : ISODate("2016-12-01T00:00:00Z"), "ToDate" :
  ↪  ISODate("2099-12-12T00:00:00Z") }
{ "_id" : ObjectId("58da897c29b4c2a
...
```

Listing A.66: Zugriff von der Lure Box auf die Honeytokens in der MongoDB-Datenbank.

A.6.7. Anbindung an die Lure Box

Konfiguration von Filebeat

```
#========================== Filebeat prospectors =============================
filebeat.prospectors:
- input_type: log

  # path to search for log data
  paths:
    - /var/lib/mongodb/auditLog.json

#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.56.254:5046"]
```

Listing A.67: Änderungen, die an der Konfigurationsdatei von Filebeat vorgenommen werden müssen.

```
[program:filebeat_mongodb]
directory=/
user=filebeat
command=/usr/share/filebeat/bin/filebeat -e
 -c /etc/filebeat/filebeat.yml
 -path.home /usr/share/filebeat
 -path.config /etc/filebeat
 -path.data /var/lib/filebeat
 -path.logs /var/log/filebeat
autostart=true
autorestart=true
stderr_logfile=/var/log/filebeat/filebeat_mongodb.stderr.log
stdout_logfile=/var/log/filebeat/filebeat_mongodb.stdout.log
```

Listing A.68: Die Supervisor-Konfigurationsdatei für das Versenden der MongoDB-Logs mittels Filebeat an die Lure Box.

Logstash

```
input {
        # use filebeat as input for this log source
        beats {
                port => 5046
```

```
5                 ssl => false
6         }
7 }
8 filter {
9         # add information about the origin of the log source
10        mutate { add_field => ["[log_source][received_at]", "%{@timestamp}"] }
11        mutate { add_field => ["[log_source][host]", "192.168.56.4"] }
12        mutate { add_field => ["[log_source][product]", "MongoDB Enterprise 3.4.3"]
            ↪  }
13
14        # split the JSON message in it's fields and append the new fields to the
            ↪  audit.object field
15        json { source => "message" target => "[audit][object]" }
16
17        # summarize fields about the source
18        mutate { rename => {"[audit][object][remote][ip]" => "[src][ip]"} }
19        mutate { rename => {"[audit][object][remote][port]" => "[src][port]"} }
20
21
22        # the name of the datbase is stored within audit.object.param.ns
23        grok { match => { "[audit][object][param][ns]" =>
            ↪  "%{WORD:database_name}%{GREEDYDATA:unimportant}" } }
24        mutate { rename => {"[database_name]" => "[audit][database][name]"} }
25
26        # summarize fields about the audit
27        mutate { rename => {"[source]" => "[audit][file][name]"} }
28        mutate { rename => {"[offset]" => "[audit][file][offset]"} }
29        mutate { rename => {"[audit][object][param][command]" => "[audit][action]"}
            ↪  }
30        mutate { rename => {"[audit][object][param][ns]" =>
            ↪  "[audit][object][namespace]"} }
31        mutate { rename => {"[audit][object][result]" =>
            ↪  "[audit][query][return_code]"} }
32        mutate { rename => {"[audit][object][atype]" => "[audit][database][atype]"}
            ↪  }
33        mutate { rename => {"[audit][object][roles]" =>
            ↪  "[audit][database][user_roles]"} }
34        mutate { rename => {"[audit][object][users]" =>
            ↪  "[audit][database][username]"} }
35
36        # use the field ts.$date as timestamp for the event
37        mutate { convert => ["[audit][object][ts][$date]" , "string"] }
38        date { match => ["[audit][object][ts][$date]", "ISO8601"] }
39
40        # remove unnecessary fields
41        mutate { remove_field =>
            ↪  ["[beat][name]","[beat][version]","[beat][hostname]",
            ↪  "[audit][object][local][port]","[audit][object][local][ip]",
            ↪  "input_type","host","[audit][object][ts][$date]","type",
            ↪  "message","unimportant"] }
```

```
}
output {
        # uncomment the following line for getting a debug output in logstash
        #stdout { codec => rubydebug}

        # write the output to the elasticsearch database
        elasticsearch {
                hosts => ["127.0.0.1:9200"]
                index => "audit_databases_mongodb-%{+YYYY.MM.dd}"
        }
}
```

Listing A.69: Die Logstash-Konfigurationsdatei */etc/logstash/conf.d/postgresql_database.conf*

```
mkdir -p /usr/share/logstash/data/mongodb
chown logstash:logstash /usr/share/logstash/data/mongodb
```

Listing A.70: Ein Datenverzeichnis für die neue Logstash-Instanz anlegen.

```
[program:logstash_mongodb]
directory=/
user=logstash
command=/usr/share/logstash/bin/logstash -f
 ↪  /etc/logstash/conf.d/mongodb_database.conf --path.data
 ↪  /usr/share/logstash/data/mongodb/
autostart=true
autorestart=true
stderr_logfile=/var/log/logstash/logstash_mongodb.stderr.log
stdout_logfile=/var/log/logstash/logstash_mongodb.stdout.log
```

Listing A.71: Die Supervisor-Konfigurationsdatei für die Verarbeitung der MongoDB-Logs mittels Logstash.

ElastAlert

```
# Rule name, must be unique
name: Monitor MongoDB

# Type of alert.
# the frequency rule type alerts when num_events events occur with timeframe time
type: any

# Index to search, wildcard supported
index: audit_databases_mongodb-*

filter:
- query:
    query_string:
      query: "audit.object.namespace:'fhstp.salary'"

query_key: "[audit][object][namespace]"

realert:
 minutes: 10
# The alert is use when a match is found
alert:
- "email"

alert_subject: "ElastAlert: Access to the database {1} detected {0}."
alert_subject_args:
- "@timestamp"
- audit.database.name

alert_text_args:
- "@timestamp"

alert_text: |
 Access to to the monitored object {0} detected.
 Connect to the Kibana webinterface http://192.168.56.254:5601 to receive more
  ↪ information.
 You can also find further information about the event below.
# (required, email specific)

email:
- "incident@fhstp.ac.at"
```

Listing A.72: Die ElastAlert-Regel */etc/elastalert/rules/monitor_mongodb.yaml* zur Alarmierung von Zugriffen auf Honeytokens.

A.7. Postfix

A.7.1. Installation und Konfiguration

```
# install Postfix
apt-get install postfix apt-get install mailutils
# change the config file /etc/postfix/main.cf
mydestination = $myhostname, postfix.chinookcorp.com, chinookcorp.com,
 ↪  student-VirtualBox, localhost.localdomain, localhost
# restart Postfix
systemctl restart postfix
```

Listing A.73: Installation und Konfiguration von Postfix unter Ubuntu 16.04.

A.7.2. Erstellen der benötigten E-Mail-Konten

```
useradd -m margaret
useradd -m steve
```

Listing A.74: Hinzufügen der zwei Honeytoken-BenutzerInnen, damit E-Mail-Nachrichten an diese gesendet werden können. [166]

A.7.3. Remote E-Mail-Versand

```
# install the Swiss Army Knife for SMTP
apt-get install swaks

# send the mails
echo "content" | swaks --to margaret@chinookcorp.com --from
 ↪  "newsletter@chinookcorp.com" --server 192.168.56.5
echo "content" | swaks --to steve@chinookcorp.com --from
 ↪  "newsletter@chinookcorp.com" --server 192.168.56.5
echo "content" | swaks --to margaret@chinookcorp.com --from "attack@insider.com"
 ↪  --server 192.168.56.5
echo "content" | swaks --to steve@chinookcorp.com --from "attack@insider.com"
 ↪  --server 192.168.56.5
```

Listing A.75: Installation von *swaks* und das Versenden von E-Mail-Nachrichten an die Honeytoken-MitarbeiterInnen von der Lure Box aus. [152]

A.7.4. Anbindung an die Lure Box

Konfiguration von Filebeat

```
#========================== Filebeat prospectors =============================
filebeat.prospectors:

- input_type: log

  # path to search for log data
  paths:
    - /var/log/mail.log

#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.56.254:5045"]
```

Listing A.76: Änderungen, die an der Konfigurationsdatei von Filebeat vorgenommen werden müssen.

```
chown syslog:filebeat /var/log/mail.log
```

Listing A.77: Dem Konto Filebeat den lesenden Zugriff auf die Logdaten von Postfix ermöglichen.

```
[program:filebeat_postfix]
directory=/
user=filebeat
command=/usr/share/filebeat/bin/filebeat -e
 -c /etc/filebeat/filebeat.yml
 -path.home /usr/share/filebeat
 -path.config /etc/filebeat
 -path.data /var/lib/filebeat
 -path.logs /var/log/filebeat
autostart=true
autorestart=true
stderr_logfile=/var/log/filebeat/filebeat_postfix.stderr.log
stdout_logfile=/var/log/filebeat/filebeat_postfix.stdout.log
```

Listing A.78: Die Supervisor-Konfigurationsdatei für das Versenden der MongoDB-Logs mittels Filebeat an die Lure Box.

Logstash

```
input {
        # use filebeat as input for this log source
        beats {
                port => 5045
                ssl => false
        }
}
filter {
        # add information about the origin of the log source
        mutate { add_field => ["[log_source][received_at]", "%{@timestamp}"] }
        mutate { add_field => ["[log_source][host]", "192.168.56.5"] }
        mutate { add_field => ["[log_source][product]", "Postfix"] }

        # split up the message into the SYSLOG format
        grok {
                patterns_dir => ["/etc/logstash/patterns"]
                match => { "message" => "%{SYSLOGBASE}
                  ↪  %{GREEDYDATA:syslog_message}" }
        }
        # split up the logline which contains the recipient address of a mail
        if [program] =~ /^postfix.*\/local$/ {
                grok {
                        patterns_dir   => "/etc/logstash/patterns"
                        match          => [ "syslog_message", "^%{POSTFIX_LOCAL}$"
                          ↪  ]
                        tag_on_failure => [ "_grok_postfix_local_nomatch" ]
                        add_tag        => [ "_grok_postfix_success" ]
                }
        }
        # split up the logline which contains the sender address of the mail send
          ↪  locally
        else if [program] =~ /^postfix.*\/pickup$/ {
                grok {
                        patterns_dir   => "/etc/logstash/patterns"
                        match          => [ "syslog_message", "^%{POSTFIX_PICKUP}$"
                          ↪  ]
                        tag_on_failure => [ "_grok_postfix_pickup_nomatch" ]
                        add_tag        => [ "_grok_postfix_success" ]
                }
        }
        # split up the logline which contains the sender address of the mail from
          ↪  external
        else if [program] =~ /^postfix.*\/qmgr$/ {
                grok {
                        patterns_dir   => "/etc/logstash/patterns"
                        match          => [ "syslog_message", "^%{POSTFIX_QMGR}$" ]
                        tag_on_failure => [ "_grok_postfix_qmgr_nomatch" ]
                        add_tag        => [ "_grok_postfix_success" ]
```

```
44                 }
45         }
46
47         # drop all other loglines of postfix
48         else { drop {} }
49
50         # split up the key-value pairs of postfix
51         if [postfix_keyvalue_data] {
52                 kv {
53                         source       => "postfix_keyvalue_data"
54                         trim_value   => "<>,"
55                         prefix       => "postfix_"
56                         remove_field => [ "postfix_keyvalue_data" ]
57                 }
58         }
59
60     # summarize fields about the received logline
61         mutate { rename => {"[postfix_relay]" => "[audit][postfix][relay]"} }
62         mutate { rename => {"[postfix_status]" => "[audit][postfix][status]"} }
63         #mutate { rename => {"[postfix_to]" => "[audit][postfix][to]"} }
64         mutate { rename => {"[postfix_uid]" => "[audit][postfix][uid]"} }
65         mutate { rename => {"[program]" => "[audit][postfix][component]"} }
66         mutate { rename => {"[source]" => "[audit][file][name]"} }
67         mutate { rename => {"[offset]" => "[audit][file][offset]"} }
68
69         # the aggregate plugin allows to aggregate multiple events which belong to
          ↪  the same task
70         # postfix logs the sender and recipient of a mail message with multiple log
          ↪  lines which
71         # belong to the same task
72         if [postfix_from] {
73                 aggregate {
74                         task_id => "%{postfix_queueid}"
75                         code => "map['from'] = event.get('postfix_from')"
76                         map_action => "create"
77                 }
78         }
79         if [postfix_to] {
80                 aggregate {
81                         task_id => "%{postfix_queueid}"
82                         code => "event.set('postfix_from',map['from'])"
83                         map_action => "update"
84                         end_of_task => true
85                         push_map_as_event_on_timeout => true
86                         timeout => 600 # 10 minutes timeout
87                 }
88                 mutate { rename => {"[postfix_to]" => "[audit][postfix][to]"} }
89                 mutate { rename => {"[postfix_from]" => "[audit][postfix][from]"} }
90                 mutate { rename => {"[postfix_queueid]" =>
                   ↪  "[audit][postfix][queue_id]"} }
```

```
91
92          }
93
94          # build the correct @timestamp field out of the timestamp field
95          # the year is not part of a postfix logline, therefore the year is taken
              ↪  out of the date when the logline was received by logstash
96          grok { match => { "@timestamp" => "%{YEAR:year}%{GREEDYDATA:unimportant}" }
              ↪  }
97          mutate { replace => ["timestamp", "%{year}%{timestamp}"] }
98          mutate { convert => ["timestamp" , "string"] }
99          date { match => ["timestamp", "yyyyMMM dd HH:mm:ss"] }
100
101         # add a TAG to the event which will be used in ElastAlert to decide if an
              ↪  alert should be created
102         if [audit][postfix][to] in
              ↪  ["margaret@chinookcorp.com","steve@chinookcorp.com"] and !
              ↪  ([audit][postfix][from] =~ /^.*@chinookcorp.com$/){
103                 mutate { add_tag => "create_alert" }
104         }
105         # remove unnecessary fields
106         mutate { remove_field =>
              ↪  ["[beat][name]","[beat][version]","[beat][hostname]","host",
              ↪  "input_type","logsource","pid","postfix_delay","postfix_delays",
              ↪  "postfix_dsn","type","unimportant","year","timestamp",
              ↪  "syslog_message","message"] }
107
108 }
109 output {
110         # uncomment the following line for getting a debug output in logstash
111         # stdout { codec => rubydebug}
112
113         # write the output to the elasticsearch database
114         elasticsearch {
115                 hosts => ["127.0.0.1:9200"]
116                 index => "audit_postfix-%{+YYYY.MM.dd}"
117         }
118 }
```

Listing A.79: Die Logstash-Konfigurationsdatei */etc/logstash/conf.d/postgresql_database.conf*

```
1 /usr/share/logstash/bin/logstash-plugin install logstash-filter-aggregate
```

Listing A.80: Installation des Logstash Aggregate Plug-ins. [155]

```
mkdir -p /usr/share/logstash/data/postfix
chown logstash:logstash /usr/share/logstash/data/postfix
```

Listing A.81: Ein Datenverzeichnis für die neue Logstash-Instanz anlegen.

```
[program:logstash_postfix]
directory=/
user=logstash
command=/usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/postfix.conf
 ↪  --path.data /usr/share/logstash/data/postfix/ -w 1
autostart=true
autorestart=true
stderr_logfile=/var/log/logstash/logstash_postfix.stderr.log
stdout_logfile=/var/log/logstash/logstash_postfix.stdout.log
```

Listing A.82: Die Supervisor-Konfigurationsdatei für die Verarbeitung der MongoDB-Logs mittels Logstash.

ElastAlert

```
# Rule name, must be unique
name: Monitor Postfix

# Type of alert.
# the frequency rule type alerts when num_events events occur with timeframe time
type: any

# Index to search, wildcard supported
index: audit_postfix-*

filter:
- query:
    query_string:
      query: "tags:'create_alert'"

query_key: "[audit][postfix][queue_id]"

realert:
 minutes: 10
# The alert is use when a match is found
alert:
- "email"

alert_subject: "ElastAlert: Possible data leak in your company detected."
alert_subject_args:
- "@timestamp"

alert_text_args:
- "audit.postfix.to"
- "audit.postfix.from"

alert_text: |
 An external mail to an internal address {0} was received.
 The internal address is only allowed to receive messages from internal senders.
 Mails from external addresses indicate data loss.
 The sender address was: {1}
 Connect to the Kibana webinterface http://192.168.56.254:5601 to receive more
  ↪  information.
 You can also find further information about the event below.

email:
- "incident@fhstp.ac.at"
```

Listing A.83: Die ElastAlert-Regel */etc/elastalert/rules/monitor_postfix.yaml* zur Alarmierung von Zugriffen auf Honeytokens.

Literaturverzeichnis

[1] T. Neaves. Have I been pwned? Pwned websites. [Accessed May 31, 2017]. [Online]. Available: https://haveibeenpwned.com/PwnedWebsites

[2] B. Krebs. As Scope of 2012 Breach Expands, LinkedIn to Again Reset Passwords for Some Users. [Accessed May 31, 2017]. [Online]. Available: http://krebsonsecurity.com/2016/05/as-scope-of-2012-breach-expands-linkedin-to-again-reset-passwords-for-some-users/

[3] C. Scott. Protecting Our Members | Official LinkedIn Blog. [Accessed May 31, 2017]. [Online]. Available: https://blog.linkedin.com/2016/05/18/protecting-our-members

[4] M. Hosenball. NSA chief says Snowden leaked up to 200,000 secret documents. [Accessed May 31, 2017]. [Online]. Available: http://www.reuters.com/article/us-usa-security-nsa-idUSBRE9AD19B20131114

[5] P. Szoldra. A timeline of Edward Snowden leaks – Business Insider Deutschland. [Accessed May 31, 2017]. [Online]. Available: http://www.businessinsider.de/snowden-leaks-timeline-2016-9?r=US&IR=T

[6] Trend Micro Incorporated. Ransomware – Definition. [Accessed May 31, 2017]. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/ransomware

[7] LegalArchiver.org. HIPAA. Health Insurance Portability and Accountability Act of 1996. [Accessed May 31, 2017]. [Online]. Available: http://www.legalarchiver.org/hipaa.htm

[8] PCI Security Standards Council. Payment Card Industry (PCI) Datensicherheitsstandard. [Accessed May 31, 2017]. [Online]. Available: https://de.pcisecuritystandards.org/_onelink_/pcisecurity/en2de/minisite/en/docs/PCI_DSS_v3.pdf

[9] LegalArchiver.org. The Sarbanes-Oxley Act 2002. [Accessed May 31, 2017]. [Online]. Available: http://www.legalarchiver.org/soa.htm

[10] S. Moore. Gartner Says Worldwide Information Security Spending Will Grow 7.9 Percent to Reach $81.6 Billion in 2016. [Accessed May 31, 2017]. [Online]. Available: http://www.gartner.com/newsroom/id/3404817

[11] Europäische Union. Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016. [Accessed May 31, 2017]. [Online]. Available: http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32016R0679&from=de

[12] L. Spitzner. Honeytokens: The Other Honeypot. [Accessed May 31, 2017]. [Online]. Available: https://www.symantec.com/connect/articles/honeytokens-other-honeypot

[13] A. Liska, *Building an Intelligence-Led Security Program*. Syngress, 2014.

[14] Department of Homeland Security. Combating the Insider Threat. [Accessed May 31, 2017]. [Online]. Available: https://www.us-cert.gov/sites/default/files/publications/Combating%20the%20Insider%20Threat_0.pdf

[15] M. D. Waters, "Identifying and Preventing Insider Threats," Master's thesis, Eastern Kentucky University, 2016.

[16] T. Werner and E. Reusch. Sondervorlesung „Netz-Sicherheit": Honeypots – Elektronische Köder im Internet. [Accessed May 31, 2017]. [Online]. Available: https://net.cs.uni-bonn.de/fileadmin/events/Special/BSI/2006-11-07_Sondervorlesung_Netzsicherheit_publication.pdf

[17] J. Strand and P. Asadoorian and E. Robishand and B. Donnelly, *Offensive Countermeasures: The Art of Active Defense*. CreateSpace Independent Publishing Platform, 2013.

[18] L. Spitzner, "Honeypots: Catching the Insider Threat," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 170–179.

[19] N. Provos. Honeypot Background. [Accessed May 31, 2017]. [Online]. Available: http://www.honeyd.org/background.php

[20] dionaea. Welcome to dionaea's documentation! [Accessed May 31, 2017]. [Online]. Available: https://dionaea.readthedocs.io/en/latest/

[21] N. Provos. Developments of the Honeyd Virtual Honeypot. [Accessed May 31, 2017]. [Online]. Available: http://www.honeyd.org/

[22] MushMush Foundation. mushorg/conpot: ICS/SCADA honeypot. [Accessed May 31, 2017]. [Online]. Available: https://github.com/mushorg/conpot

[23] J. Nazario. Awesome Honeypots. [Accessed May 31, 2017]. [Online]. Available: https://github.com/paralax/awesome-honeypots

[24] M. Milton. Honeytokens: The State of the Art. [Accessed May 31, 2017]. [Online]. Available: https://www.amazon.com/Honeytokens-State-Art-Mark-Milton-ebook/dp/B00BKYRAE4

[25] P. Sokol and M. Husak and F. Lipták, "Deploying Honeypots and Honeynets: Issue of Privacy," in *2015 10th International Conference on Availability, Reliability and Security*, Aug 2015, pp. 397–403.

[26] SecurityFocus. SecurityFocus. [Accessed May 31, 2017]. [Online]. Available: http://www.securityfocus.com/archive/119

[27] R. Petrunić, "Honeytokens as active defense," in *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1313–1317.

[28] A. Shabtai and M. Bercovitch and L. Rokach and Y. Gal and Y. Elovici and E. Shmueli, "Behavioral Study of Users When Interacting with Active Honeytokens," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 3, pp. 9:1–9:21, feb 2016.

[29] K. Padayachee, "Aspectizing Honeytokens to Contain the Insider Threat," *IET Information Security*, vol. 9, no. 4, pp. 240–247, 2014.

[30] T. Holt, *The Deceivers: Allied Military Deception in the Second World War*. Simon and Schuster, 2010.

[31] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. Simon and Schuster, 2005.

[32] UK Essays. Advantages And Disadvantages Involved In Honeytokens Information Technology Essay. [Accessed May 31, 2017]. [Online]. Available: https://www.ukessays.com/essays/information-technology/advantages-and-disadvantages-involved-in-honeytokens-information-technology-essay.php

[33] B. Bowen and S. Hershkop and A. Keromytis and S. J. Stolfo, "Baiting Inside Attackers Using Decoy Documents," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2009, pp. 51–70.

[34] A. Shabtai and Y. Elovici and L. Rokach, *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media, 2012.

[35] M. Bercovitch and M. Renford and L. Hasson and A. Shabtai and L. Rokach and Y. Elovici, "HoneyGen: an Automated Honeytokens Generator," in *Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics*, July 2011, pp. 131–136.

[36] Black Hills Information Security. ADHD – Black Hills Information Security. [Accessed May 31, 2017]. [Online]. Available: http://www.blackhillsinfosec.com/?page_id=4419

[37] ——. ADHD Tools Usage Document. [Accessed May 31, 2017]. [Online]. Available: https://adhdproject.github.io/

[38] A. Ka. Honeybits. [Accessed May 31, 2017]. [Online]. Available: https://github.com/0x4D31/honeybits

[39] Thinkst Canary. Canarytokens by Thinkst. [Accessed May 31, 2017]. [Online]. Available: http://canarytokens.org/

[40] ——. Canarytokens helps track activity and actions on your network. [Accessed May 31, 2017]. [Online]. Available: https://github.com/thinkst/canarytokens

[41] ——. Canarytokens.org – Quick, Free, Detection for the Masses. [Accessed May 31, 2017]. [Online]. Available: http://blog.thinkst.com/p/canarytokensorg-quick-free-detection.html

[42] SecureWorks. secureworks/dcept: A tool for deploying and detecting use of Active Directory honeytokens. [Accessed May 31, 2017]. [Online]. Available: https://github.com/secureworks/dcept

[43] ——. DCEPT: An Open-Source Honeytoken Tripwire. [Accessed May 31, 2017]. [Online]. Available: https://www.secureworks.com/blog/dcept

[44] O. Zohar and A. Barbalat and R. Elharar, "Applying Deception Mechanisms for Detecting Sophisticated Cyber Attacks," TopSpin, Tech. Rep., 2016.

[45] J. Yuill and M. Zappe and D. Denning and F. Feer, "Honeyfiles: Deceptive Files for Intrusion Detection," in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, June 2004, pp. 116–122.

[46] N. Virvilis and B. Vanautgaerden and O. S. Serrano, "Changing the game: The art of deceiving sophisticated attackers," in *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, June 2014, pp. 87–97.

[47] T. Neaves. Honey Tokens, The Universe and Everything. [Accessed May 31, 2017]. [Online]. Available: https://web.archive.org/web/20090220062427/http://www.tomneaves.co.uk/2006/07/honey-tokens-the-universe-and-everything.html

[48] A. Čenys and D. Rainys and L. Radvilavicius and N. Goranin, "Database Level Honeytoken Modules for Active DBMS Protection," in *Advances in Information Systems Development*. Springer, 2006, pp. 449–457.

[49] H. AbdelallahElhadjº and H. M. Khelalfa and H. M. Kortebi, "An experimental sniffer detector: SnifferWall," *Atelier SEcurité des Communications sur Internet (SECI'02) Hôtel El Mechtel*, pp. 69–80, 2002.

[50] S. V. Reddy and K. S. Ramani and K. Rijutha and S. M. Ali and C. P. Reddy, "Wireless hacking – a WiFi hack by cracking WEP," in *2010 2nd International Conference on Education Technology and Computer*, vol. 1, June 2010, pp. V1–189–V1–193.

[51] J. Cossu and H. ElAarag, "Malware Detection Inspired by the Human Immune System and Making Use of Honeytokens," in *Proceedings of the 18th Symposium on Communications & Networking*, ser. CNS '15. Society for Computer Simulation International, 2015, pp. 37–43.

[52] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 145–160.

[53] M. Rouse. What is security information and event management (SIEM)? [Accessed May 31, 2017]. [Online]. Available: http://searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM

[54] Splunk Inc. Operational Intelligence, Log Management, Application Management, Enterprise Security and Compliance | Splunk. [Accessed May 31, 2017]. [Online]. Available: https://www.splunk.com/

[55] Hewlett Packard Enterprise Development LP. SIEM, Enterprise Security Information Event Management Solutions | Hewlett Packard Enterprise. [Accessed May 31, 2017]. [Online]. Available: https://saas.hpe.com/en-us/software/siem-security-information-event-management

[56] International Business Machines Corp. IBM Security QRadar SIEM. [Accessed May 31, 2017]. [Online]. Available: http://www-03.ibm.com/software/products/en/qradar-siem

[57] Elasticsearch. Overview | Filebeat Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/5.3/filebeat-overview.html

[58] ——. Logstash Introduction | Logstash Reference [5.3]. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/introduction.html

[59] ——. Getting Started | Elasticsearch Reference [5.3]. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/5.3/getting-started.html

[60] ——. Introduction | Kibana User Guide [5.3]. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/kibana/5.3/introduction.html

[61] Yelp Inc. Yelp/elastalert: Easy & Flexible Alerting With ElasticSearch. [Accessed May 31, 2017]. [Online]. Available: https://github.com/Yelp/elastalert

[62] L. Rocha. Chinook Database. [Accessed May 31, 2017]. [Online]. Available: https://chinookdatabase.codeplex.com/

[63] ——. Chinook Schema. [Accessed May 31, 2017]. [Online]. Available: https://chinookdatabase.codeplex.com/wikipage?title=Chinook_Schema&referringTitle=Documentation

[64] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, jun 1970.

[65] C. Birgen, "SQL vs. NoSQL," Norwegian University of Science and Technology, Tech. Rep., 2014.

[66] P. Näsholm, "Extracting Data from NoSQL Databases – A Step towards Interactive Visual Analysis of NoSQL Data," Master's thesis, Chalmers University of Technology, 2012.

[67] Wikipedia. Begriffe relationaler Datenbanken – Relationale Datenbank. [Accessed May 31, 2017]. [Online]. Available: https://de.wikipedia.org/wiki/Relationale_Datenbank#/media/File:Begriffe_relationaler_Datenbanken.svg

[68] C. Strauch. NoSQL Databases. [Accessed May 31, 2017]. [Online]. Available: http://www.christof-strauch.de/nosqldbs.pdf

[69] W. da Rocha Franca, *MongoDB Data Modeling*. Packt Publishing, 2015.

[70] Oracle Corporation. Database Security Guide – Introduction to Auditing. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/DBSEG/auditing.htm#DBSEG1023

[71] K. Simmons and S. Carstarphen, *Pro SQL Server 2012 Administration*. Apress, 2012.

[72] D. Fabbri and R. Ramamurthy and R. Kaushik, "SELECT triggers for data auditing," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, April 2013, pp. 1141–1152.

[73] Oracle Corporation. Oracle VM VirtualBox. [Accessed May 31, 2017]. [Online]. Available: https://www.virtualbox.org/

[74] ——. Chapter 6. Virtual networking. [Accessed May 31, 2017]. [Online]. Available: https://www.virtualbox.org/manual/ch06.html#network_hostonly

[75] Agendaless Consulting and Contributors. Supervisor 3.3.1 documentation. [Accessed May 31, 2017]. [Online]. Available: http://supervisord.org/introduction.html

[76] Elasticsearch. Repositories for APT and YUM | Filebeat Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/5.3/setup-repositories.html

[77] ——. Basic Concepts | Elasticsearch Reference [5.3]. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/5.3/_basic_concepts.html

[78] L. Menezes. lmenezes/cerebro. [Accessed May 31, 2017]. [Online]. Available: https://github.com/lmenezes/cerebro

[79] Elasticsearch. Stashing Your First Event | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/first-event.html

[80] ——. Installing Logstash | Logstash Reference [5.3]] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/installing-logstash.html

[81] ——. How Logstash Works | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/pipeline.html

[82] ——. Input plugins | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/input-plugins.html

[83] ——. Filter plugins | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/filter-plugins.html

[84] ——. csv | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-csv.html

[85] ——. date | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-date.html

[86] ——. drop | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-drop.html

[87] ——. grok | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-grok.html

[88] ——. json | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-json.html

[89] ——. mutate | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-mutate.html

[90] ——. Logstash Configuration Examples | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/config-examples.html

[91] ——. Running Logstash from the Command Line | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/running-logstash-command-line.html

[92] ——. Install Kibana with Debian Package | Kibana User Guide [5.3]. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/kibana/5.3/deb.html

[93] ——. Security: Enterprise Security for Elasticsearch | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/products/x-pack/security

[94] ——. Alerting: Alerts & Notifications for Elasticsearch. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/products/x-pack/alerting

[95] Yelp Inc. Running ElastAlert for the First Time. [Accessed May 31, 2017]. [Online]. Available: https://elastalert.readthedocs.io/en/latest/running_elastalert.html

[96] Oracle Corporation. Introduction to Oracle Database. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/CNCPT/intro.htm#CNCPT88784

[97] solid IT gmbh. DB-Engines Ranking – popularity ranking of database management systems. [Accessed May 01, 2017]. [Online]. Available: http://db-engines.com/en/ranking

[98] Oracle Corporation. Oracle Technology Network Developer Day – Database Virtual Box Appliance / Virtual Machine. [Accessed May 31, 2017]. [Online]. Available: http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html

[99] M. A. Miller and S. Kost, "Oracle 12c Unified Auditing," Integrigy Corporation, Tech. Rep., 2016.

[100] T. Salama. Oracle Database 12c Security: New Unified Auditing. [Accessed May 31, 2017]. [Online]. Available: https://blogs.oracle.com/imc/entry/oracle_database_12c_new_unified

[101] Oracle Corporation. Database Security Guide – Introduction to Auditing. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/DBSEG/audit_admin.htm#DBSEG1026

[102] ——. Database Security Guide – Configuring Audit Policies. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/DBSEG/audit_config.htm#DBSEG1025

[103] A. Čenys and D. Rainys and L. Radvilavicius and A. Bielko, "Development of Honeypot System Emulating Functions of Database Server," Vilnius Gediminas Technical University, Tech. Rep., 2004.

[104] L. L. Ray and H. Felch, "Methodology for Detecting Advanced Persistent Threats in Oracle Databases," *International Journal of Strategic Information Technology and Applications (IJSITA)*, vol. 5, no. 1, pp. 42–53, 2014.

[105] Oracle Corporation. Database Upgrade Guide – Post-Upgrade Tasks for Oracle Database. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/UPGRD/afterup.htm#UPGRD004

[106] ——. Database Security Guide – Administering the Audit Trail. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/DBSEG/audit_admin.htm#DBSEG631

[107] ——. Oracle Instant Client Downloads. [Accessed May 31, 2017]. [Online]. Available: http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html

[108] ——. Database Security Guide – Administering the Audit Trail. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/database/121/DBSEG/audit_admin.htm#DBSEG733

[109] ——. Oracle Database 12.2.0.1 JDBC Driver & UCP Downloads. [Accessed May 31, 2017]. [Online]. Available: http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html

[110] A. Jorgensen and J. Segarra and P. LeBlanc, *Microsoft SQL Server 2012 Bible*. John Wiley & Sons, 2012.

[111] A. Wolter and K. Höltgen and F. Geisler and C. Klein, *SQL Server 2014: Ein Blick in die Datenzukunft*. entwickler.Press, 2014.

[112] Microsoft Corporation. SQL Server Management Studio (SSMS). [Accessed May 31, 2017]. [Online]. Available: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms

[113] ——. sqlcmd Utility. [Accessed May 31, 2017]. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms162773.aspx

[114] R. Byham and C. Guyer. SQL Server Audit (Database Engine). [Accessed May 31, 2017]. [Online]. Available: https://msdn.microsoft.com/en-us/library/cc280386.aspx

[115] ——. SQL Server Audit (Database Engine). [Accessed May 31, 2017]. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/auditing/sql-server-audit-database-engine

[116] ——. CREATE SERVER AUDIT SPECIFICATION (Transact-SQL). [Accessed May 31, 2017]. [Online]. Available: https://msdn.microsoft.com/de-at/library/cc280448.aspx

[117] Microsoft Corporation. Prinzipale (Datenbankmodul). [Accessed May 31, 2017]. [Online]. Available: https://msdn.microsoft.com/de-at/library/ms181127.aspx

[118] J. Woo and S. Lee and C. Zoltowski, “Database Auditing.”

[119] R. Byham and C. Guyer. SQL Server Utilities Statements – GO. [Accessed May 31, 2017]. [Online]. Available: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/sql-server-utilities-statements-go

[120] Microsoft Corporation. 1.3 Overview. [Accessed May 31, 2017]. [Online]. Available: https://msdn.microsoft.com/en-us/library/dd357628.aspx

[121] FreeTDS Project. FreeTDS.org. [Accessed May 31, 2017]. [Online]. Available: http://www.freetds.org/index.html

[122] Microsoft Corporation. sys.fn_get_audit_file (Transact-SQL). [Accessed May 31, 2017]. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/system-functions/sys-fn-get-audit-file-transact-sql

[123] ——. Download Microsoft JDBC Driver for SQL Server. [Accessed May 31, 2017]. [Online]. Available: https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server

[124] PostgreSQL Global Development Group. PostgreSQL: About. [Accessed May 31, 2017]. [Online]. Available: https://www.postgresql.org/about/

[125] ubuntu Deutschland e. V. PostgreSQL. [Accessed May 31, 2017]. [Online]. Available: https://wiki.ubuntuusers.de/PostgreSQL

[126] J. Ellingwood. How To Install and Use PostgreSQL on Ubuntu 16.04. [Accessed May 31, 2017]. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-16-04

[127] PostgreSQL Development Community. PostgreSQL Auditing Extension | PGAudit. [Accessed May 31, 2017]. [Online]. Available: http://pgaudit.org/

[128] ——. pgAudit Open Source PostgreSQL Audit Logging. [Accessed May 31, 2017]. [Online]. Available: https://github.com/pgaudit/pgaudit/blob/master/README.md

[129] PostgreSQL Global Development Group. PostgreSQL: Documentation: 9.6: Error Reporting and Logging. [Accessed May 31, 2017]. [Online]. Available: https://www.postgresql.org/docs/9.6/static/runtime-config-logging.html

[130] C. Chauhan, *PostgreSQL Cookbook*. Packt Publishing, 2015.

[131] PostgreSQL Global Development Group. PostgreSQL: Documentation: 9.6: The pg_hba.conf File. [Accessed May 31, 2017]. [Online]. Available: https://www.postgresql.org/docs/9.6/static/auth-pg-hba-conf.html

[132] T. Erik and P. Brown. logrotate. [Accessed May 31, 2017]. [Online]. Available: http://www.linuxcommand.org/man_pages/logrotate8.html

[133] Elasticsearch. Step 2: Configuring Filebeat | Filebeat Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/5.3/filebeat-configuration.html

[134] ——. beats | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-inputs-beats.html

[135] PostgreSQL Global Development Group. PostgreSQL: Documentation: 9.6: Error Reporting and Logging. [Accessed May 31, 2017]. [Online]. Available: https://www.postgresql.org/docs/9.6/static/runtime-config-logging.html

[136] P. Murugesan and I. Ray, "Audit Log Management in MongoDB," in *2014 IEEE World Congress on Services*, June 2014, pp. 53–57.

[137] E. Plugge and D. Hows and P. Membrey and T. Hawkins, *The Definitive Guide to MongoDB: A complete guide to dealing with Big Data using MongoDB*. Apress, 2015.

[138] MongoDB Inc. Install MongoDB Enterprise on Ubuntu. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-ubuntu/

[139] ——. Import Example Dataset. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/getting-started/shell/import-data/

[140] ——. Auditing – MongoDB Manual 3.4. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/manual/core/auditing/

[141] Percona LLC. Percona Server for MongoDB. [Accessed May 31, 2017]. [Online]. Available: https://www.percona.com/software/mongo-database/percona-server-for-mongodb

[142] M. Cory, "MongoDB Incidence Response," Master's thesis, University of Central Florida, United States, Florida, 2016.

[143] MongoDB Inc. Configure Auditing – MongoDB Manual 3.4. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/master/tutorial/configure-auditing/

[144] ——. Database Profiler – MongoDB Manual 3.4. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/manual/tutorial/manage-the-database-profiler/

[145] ——. mongoreplay – MongoDB Manual 3.4. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/manual/reference/program/mongoreplay/

[146] ——. Enable Auth – MongoDB Manual 3.4. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/manual/tutorial/enable-authentication/

[147] ——. System Event Audit Messages – MongoDB Manual 3.2. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/v3.2/reference/audit-message/#audit-action-details-results

[148] K. D. Dent, *Postfix: The Definitive Guide*. O'Reilly Media, 2003.

[149] Postfix Projekt. The Postfix Home Page. [Accessed May 31, 2017]. [Online]. Available: http://www.postfix.org/start.html

[150] Canonical Ltd. Ubuntu – Details of package postfix in xenial. [Accessed May 31, 2017]. [Online]. Available: http://packages.ubuntu.com/xenial/postfix

[151] Postfix Projekt. Postfix Basic Configuration. [Accessed May 31, 2017]. [Online]. Available: http://www.postfix.org/BASIC_CONFIGURATION_README.html

[152] J. Jetmore. Swaks – Swiss Army Knife for SMTP. [Accessed May 31, 2017]. [Online]. Available: http://www.jetmore.org/john/code/swaks/

[153] H. Morikawa. logstash. [Accessed May 31, 2017]. [Online]. Available: https://github.com/nxhack/logstash/blob/master/patterns/postfix

[154] T. Hendrikx. Logstash grok patterns for postfix logging. [Accessed May 31, 2017]. [Online]. Available: https://github.com/whyscream/postfix-grok-patterns

[155] Elasticsearch. aggregate | Logstash Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/logstash/5.3/plugins-filters-aggregate.html

[156] ——. Powering Data Search, Log Analysis, Analytics | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/products

[157] The Apache Software Foundation. Apache Cassandra. [Accessed May 31, 2017]. [Online]. Available: http://cassandra.apache.org/

[158] Neo Technology Inc. Neo4j, the world's leading graph database. [Accessed May 31, 2017]. [Online]. Available: https://neo4j.com/

[159] The Apache Software Foundation. Documentation. [Accessed May 31, 2017]. [Online]. Available: http://cassandra.apache.org/doc/4.0/

[160] Neo Technology Inc. The Neo4j Operations Manual v3.1. [Accessed May 31, 2017]. [Online]. Available: https://neo4j.com/docs/operations-manual/3.1/

[161] Elasticsearch. Install Elasticsearch with Debian Package | Elasticsearch Reference [5.3] | Elastic. [Accessed May 31, 2017]. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/5.3/deb.html

[162] Oracle. JDK Installation for Linux Platforms. [Accessed May 31, 2017]. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/install/linux_jdk.html#BJFJJEFG

[163] A. E. S. Lourenço. Elastalert: implementing rich monitoring with Elasticsearch. [Accessed May 31, 2017]. [Online]. Available: https://alexandreesl.com/2016/04/15/elastalert-implementing-rich-monitoring-with-elasticsearch/

[164] C. Cristea. Installing Oracle SQL*Plus client on Ubuntu. [Accessed May 31, 2017]. [Online]. Available: http://webikon.com/cases/installing-oracle-sql-plus-client-on-ubuntu

[165] MongoDB Inc. Install MongoDB Enterprise on Debian – MongoDB Manual 3.4. [Accessed May 31, 2017]. [Online]. Available: https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-debian/

[166] J. Brockmeier. Adding Users and Aliases for Postfix. [Accessed May 31, 2017]. [Online]. Available: http://www.serverwatch.com/tutorials/article.php/3912056/Adding-Users-and-Aliases-for-Postfix.htm

Abbildungsverzeichnis

Tabellenverzeichnis

Listingsverzeichnis