Intelligentes Monitoring der IT-Sicherheit durch den Einsatz von SIEM

Kai-Oliver Detken[1] · Carsten Elfers[2] · Malte Humann[3] Marcel Jahnke[1] · Stefan Edelkamp[3]

[1]DECOIT GmbH, Fahrenheitstr. 9, D-28359 Bremen
detken@decoit.de

[2]neusta GmbH, Konsul-Schmidt-Straße 24, D-28217 Bremen
c.elfers@neusta.de

[3]Universität Bremen, Technologie-Zentrum Informatik (TZI), Bibliothekstraße 1, D-28359 Bremen, mhumann/edelkamp@tzi.de

Zusammenfassung

SIEM- und Monitoring-Systeme sind Überwachungsprogramme für Rechnernetzinfrastrukturen, die Ereignismeldungen aus verschiedenen Sensoren sammeln und zusammenführen. Auf diese Weise werden sicherheitsrelevante Informationen den Verantwortlichen aus den Bereichen IT, Informationssicherheit und Management zentralisiert zur Verfügung gestellt. Die Nutzer eines SIEM-Systems innerhalb einer Firma erhalten somit einen für sie relevanten Überblick über den Sicherheitszustand des Unternehmensnetzes. Die wesentliche Aufgabe eines SIEM-Systems ist die Korrelation der heterogenen Sensorereignisse, um dadurch dem Anwender nur die für ihn relevanten Meldungen zu präsentieren. Das System iMonitor bietet eine neue Form der Ereigniskorrelation, die neue Angriffsvarianten erkennt. Durch den Einsatz diverser Sensoren steht eine große Datenbasis zur Verfügung, die mittels Methoden der KI ausgewertet und direkt in für den Menschen verständliche Handlungsempfehlungen umgesetzt werden kann. Als Basis für die Entwicklungsarbeiten wurde das Monitoring-Werkzeug Icinga um SIEM-Funktionalität erweitert.

1 Einleitung

Security Information and Event Management (SIEM) Systeme ermöglichen durch das Sammeln von Sensorinformationen und Ereignissen, Bedrohungen zu erkennen und zu verhindern [1]. So kann es beispielsweise eine organisatorische Richtlinie geben, die besagt, dass Viren, die von extern oder intern im Netzwerk versendet werden können, nicht das Server-Netz, die DMZ oder das Internet erreichen dürfen.

Wesentliche Errungenschaft von *iMonitor* sind die Entwicklung eines SIEM-Systems zur intelligenten Überwachung der IT-Sicherheit eines Unternehmens, die effiziente Erkennung von Angriffsarten und die automatische Generierung von verständlichen Handlungsempfehlungen. Funktional arbeitet *iMonitor* daher ähnlich zu SIEM-Systemen: Im Wesentlichen werden Vorgänge und Ereignisse (Firewall-Logs, Datenbank-Logs, Intrusion-Detection-Systeme) in

Firmennetzen überwacht. Wo es sinnvoll erscheint, leitet *iMonitor* automatisiert, in Echtzeit und pro-aktiv Maßnahmen zur Verbesserung der IT-Sicherheit ein.

Um sicherheitsrelevante Ereignisse (engl. Events) unterschiedlicher Güte in einem SIEM zusammenzuführen, erbringen sogenannte Kollektoren folgende Leistungen:

a. **Extraktion:** Ereignisse sind in Rohform zumeist Einträge in Log-Dateien oder über das Netz versendete Systemmeldungen. Diese Informationen werden aus den jeweiligen verwendeten Systemen oder Transportprotokollen extrahiert, um sie einem SIEM-System zugänglich zu machen.

b. **Homogenisierung/Mapping:** Ereignisse werden von unterschiedlichen Diensten erzeugt (Intrusion-Detection-Systeme) und aus unterschiedlichen Systemen extrahiert (Logs). Um eine Weiterverarbeitung in einem SIEM-System zu gewährleisten, werden die relevanten Inhalte der einzelnen Events miteinander in Bezug gebracht. Dafür sorgt ein entsprechendes "Umsortieren" individueller Datenfelder aus speziellen Ereignisformaten in ein standardisiertes, dem SIEM-System verständliches, Ereignisformat (z.B. IDMEF nach RFC-4765).

c. **Aggregation:** Große Mengen gleichartiger Ereignisse würden über einen kurzen Zeitraum ein zentrales SIEM-System belasten, ohne einen signifikanten Mehrwert zu erzeugen. Kollektoren aggregieren daher große Mengen gleichartiger Ereignisse über einen kurzen Zeitraum (sog. Bursts) zu einem einzigen Ereignis mit höherer Aussagekraft (z.B. Ereignis-Typ, Inhalt und Menge der ursprünglichen Meldungen).

Die mit der Verbreitung und der Pflege von Kollektoren verbundenen Kosten sind ein weiterer wesentlicher Grund, dass SIEM-Systeme in kleinen und mittelständigen Unternehmen (KMUs) nicht zum Einsatz kommen. SIEM-Systeme sind bereits heute sehr leistungsfähig, auf große Ereignisströme skalierbar und verwenden, zumindest teilweise, Verfahren der Künstlichen Intelligenz (KI). Diese Entwicklung führt jedoch nicht automatisch dazu, dass SIEM-Systeme für KMUs an Attraktivität gewinnen.

Um den Einführungs- und Betriebsaufwand für ein KMU auf eine annehmbare Schwelle zu senken, kombiniert *iMonitor* einfachere Verfahren, die entweder bereits von existierenden IT-Systemen, Betriebssystemen, Anwendungen und Netzen bereitgestellt werden oder mit minimalem Aufwand zusätzlich installiert werden können.

Die Auswertung von Ereignissen wird anhand von Regelsätzen durchgeführt. Dies hat den Nachteil, dass auch nur bekannte Ereignisse erkannt werden können. LogRhythm z.B. liefert bei der Auslieferung eine Vielzahl von Regeln mit und zusätzlich können eigene Regeln erstellt werden. Die Relevanz der Ergebnisse der Regelauswertung ist abhängig von den Eigenschaften bzw. dem Aufbau des jeweiligen Unternehmens und im Falle von selbsterstellten Regeln deren Genauigkeit. Beispiele hierfür sind primäre und sekundäre Geschäftsprozesse, organisatorische Prozesse, die Bedrohungslage oder eingesetzte IT-Assets. *iMonitor* verfolgt neben Regelbasierter Auswertung einen weiteren Ansatz. Nämlich das Erkennen von Anomalien. Hier wird bei der Inbetriebnahme eine gewisse Zeit der Normverkehr des Netzes aufgezeichnet. Weicht später von diesem Normverhalten der Netzverkehr ab, so wird dies als Anomalie erkannt. Dabei werden allerdings auch gewisse Abweichungen toleriert, um die False-Positve-Rate zu verringern.

Die Operationalisierung übergreifender Strategien, aber auch bereits das Ableiten von Regelsätzen aus konkreten Sicherheitsrichtlinien, stellt für KMUs eine große Herausforderung

dar. Maschinenlesbare Sicherheitsrichtlinien sind komplex und deren manuelle Erstellung erfordert spezifisches Expertenwissen, das oft nur in begrenztem Maße zur Verfügung steht. Ohne eine wirksame Menge an Regelsätzen ist ein SIEM-System in seiner Wirkung stark eingeschränkt und erbringt nicht die den Anschaffungs- und Betriebskosten entsprechende Leistung. Um die Wirksamkeit von Regelsätzen auf die Unternehmensziele auszurichten, müssen Verfahren zur Verfügung gestellt werden, die es einem KMU ermöglichen, Regelsätze basierend auf dem Zustand der aktuellen IT-Infrastruktur herzuleiten. Gerade die verständliche Darstellung von formalisierten SOLL-Zuständen ist eine notwendige Grundlage für das Erzeugen von Regeln, die Abweichungen vom IST-Zustand mit Hilfe eines SIEM-Systems erkennen sollen.

2 iMonitor-Architektur

Als Basissystem wurde die Monitoring-Lösung *Icinga* (https://www.icinga.org) ausgewählt, da diese eine Open-Source-Basis besitzt und ein detailliertes Eskalationsmanagement beinhaltet. Auch bietet es eine Vielzahl von Plug-Ins, um Erweiterungen einbetten zu können, ohne den Kern(el) verändern zu müssen. Für die Erweiterung von *Icinga* gibt es daher zwei Möglichkeiten:

a. Es können Plug-Ins entwickelt werden

b. *Icinga* selbst wird weiterentwickelt

Icinga bot bisher noch keine SIEM-Merkmale an und ist ausschließlich auf die Überwachung der Verfügbarkeit von Diensten und Systemen ausgerichtet. Für die Wahl von *Icinga* sprachen aber die sehr gute Dokumentation und die einfache Erweiterbarkeit durch Plug-Ins. Dadurch war es möglich, den Quellcode direkt anzupassen, falls die gewünschte Funktionalität nicht schon über ein Add-On oder Plug-In realisiert wurde. Zudem gab es im SIEM-Umfeld wenige offene Lösungen, auf die man hätte aufsetzen können.

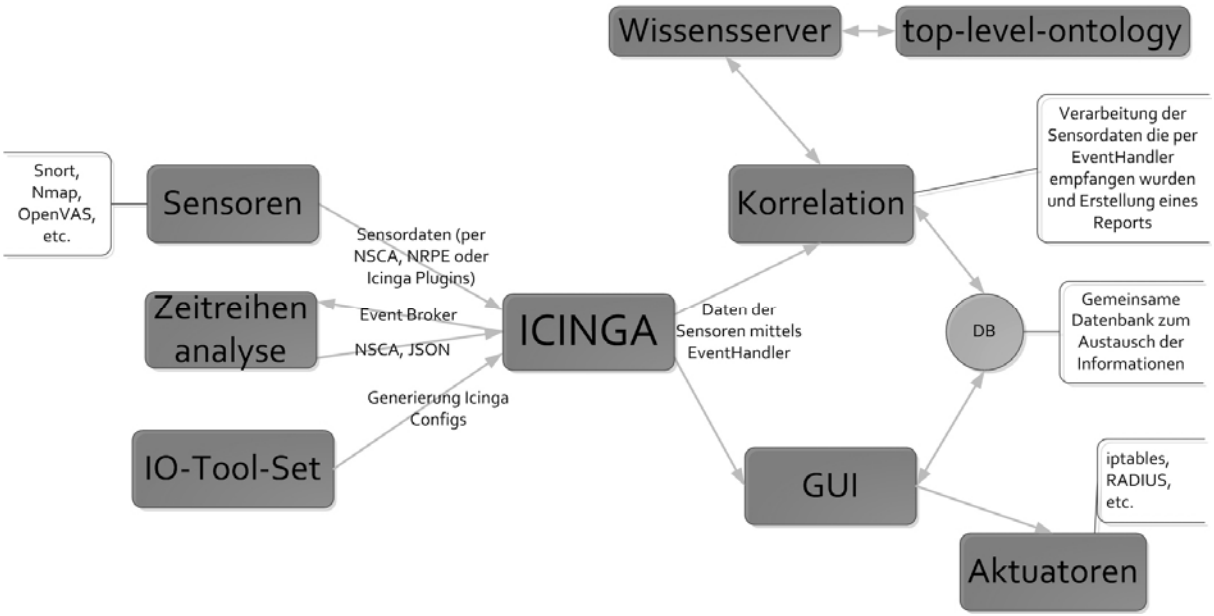

Abb. 1: SIEM-Architektur in *iMonitor*

Icinga legt alle Konfigurations- und Ereignisdaten in einer Datenbank ab. Dies hat den Vorteil, dass Daten schneller abgerufen und verarbeitet werden können. Um Daten zu speichern, nutzt *Icinga* das Add-On *IDOUtils*. Für das Abrufen von Daten stellt *Icinga* die *Icinga-Web REST-API* bereit. Viele Add-Ons nutzen diese Möglichkeit für die Verarbeitung oder Visualisierung von Daten. Als Datenbanksysteme werden bisher MySQL, Oracle und PostgreSQL unterstützt.

Abb. 1 zeigt die SIEM-Architektur von *iMonitor*. Sie stellt Icinga in den Mittelpunkt, inkl. der darunterliegenden Datenbank. Auf die Datenbank greifen eine Reihe von Systemen zu, die die folgenden Funktionen bereithalten:

a. Sensoren: Analysetools wie *Snort*, *Nmap* uns *OpenVAS* sammeln Daten der Unternehmensumgebung, um das Normalverhalten zu erkennen.

b. Zeitreihenanalyse: Analysiert das Normalverhalten und versucht Anomalien ausfindig zu machen, ohne sich allein auf Mustererkennung zu stützen.

c. IO-Toolset: Erhebt die IT-Infrastruktur und ermöglicht logische Schlüsse auf der bestehende Datenstruktur und der *Icinga*-Konfiguration.

d. GUI: Grafische Oberfläche des SIEM-Moduls, um Vorgänge, Ereignisse und Tickets sowie eine Risikoabschätzung zentral sowie übersichtlich darzustellen.

e. Korrelation: Verarbeitung der Sensorereignisdaten und Erstellung eines Berichts sowie Vorschlag von Handlungsempfehlungen.

f. **Aktuatoren:** Komponenten wie *Iptables* oder *RADIUS*, die in Echtzeit die Veränderung des Regelwerks vornehmen, stellen in einem Ticketsystem Handlungsempfehlungen dar.

3 Zeitreihenbasierte Anomalie-Erkennung

Die Zeitreihenanalyse erweitert *Icinga* um eine Komponente, die es ermöglicht, alle eingehenden *Performance-Daten* auf Anomalien zu überprüfen.[1] *Icinga* selbst ist zwar bereits in der Lage, die Daten über vorab definierte Schwellenwerte als normal, als eine Warnung oder als kritisch einzustufen, aber Anomalien müssen nicht zwangsweise diese Schwellenwerte überschreiten. So kann eine Anomalie in dem akzeptablen Wertebereich liegen und nur zu einem unüblichen Zeitpunkt auftreten. Auch das Fehlen von Werten in einem bestimmten Zeitraum kann eine Anomalie darstellen, die nicht durch einfache Schwellenwerte erkannt wird, wie in Abb. 2 KW 16-17 zu sehen. Weiterhin müssen passende Schwellenwerte vorab konfiguriert werden, um diese Überprüfung nutzen zu können.

Um den Konfigurationsaufwand möglichst gering zu halten, ermittelt die zeitreihenbasierte Anomalie-Erkennung daher die zur Auswertung nötigen Informationen selbst, auf Basis der eingehenden Performance-Daten. Die Grundlage bildet dabei das Erkennen von sich wiederholenden Mustern in den Daten. Da ein Muster mehrfach in einer Datenreihe auftauchen muss, um als solches identifiziert zu werden, benötigt das Verfahren, abhängig von der Musterlänge, entsprechend viel Vorlauf. Umgekehrt bedeutet das, dass zu Beginn der Zeitreihenanalyse, wenn nur wenige Daten bekannt sind, es vermehrt zu Fehleinschätzungen kommen kann. Bei sehr stark ausgeprägten Mustern kann die Musterlänge bereits zu Beginn des

[1] Da die Performance-Daten (http://docs.icinga.org/latest/de/perfdata.html) für *Icinga* optional sind, werden entsprechend nur solche Plug-Ins unterstützt, die diese Daten auch an *Icinga* senden.

dritten Auftretens erkannt werden. Im Fall des Beispiels aus Abb. 2 wechselt die erkannte Länge zwischen ein und sieben Tagen und stabilisiert sich erst nach etwa 6-7 Wiederholungen auf die Länge von einer Woche. Ist ein Muster bzw. die Länge des Musters gefunden, kann es verwendet werden, um den aktuellen Messwert vorherzusagen. Dieser Vorhersagewert kann dann mit dem tatsächlich an *Icinga* gemeldeten Wert verglichen werden, um ihn als normal oder Anomalie einzustufen. Anschließend wird eine entsprechende Meldung an *Icinga* geschickt, die es angeschlossenen Plug-Ins wiederum erlaubt, das Ergebnis der Anomalie-Erkennung weiterzuverwenden.

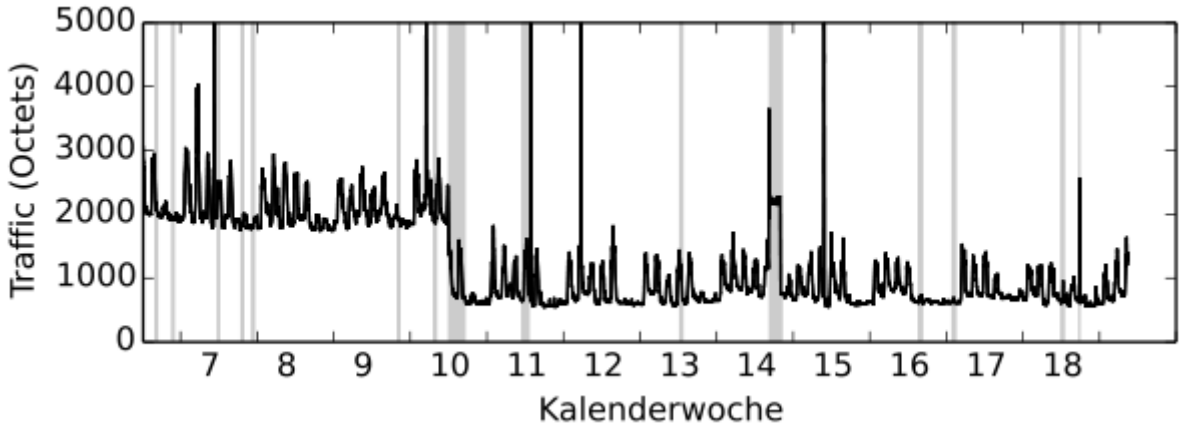

Abb. 2: Von der Zeitreihenanalyse an *Icinga* gemeldete Anomalien (grau) in durch *Icinga* via SNMP gesammelten Trafficdaten (schwarz).

Da die Performance-Daten selbst nur Zahlenwerte sind, lassen sie sich zusammen mit dem jeweiligen Zeitpunkt der Meldung als eine Zeitreihe auffassen. Um ein wiederkehrendes Muster in einer solchen Zeitreihe zu erkennen wird eine leicht modifizierte Version des *autoperiod*-Ansatzes von Vlachos et al. [VYC05] verwendet. Sie kombinieren die Informationen eines *Periodogramms* mit denen der zyklischen *Autokorrelation*, um die jeweiligen Schwächen auszugleichen. Das Periodogramm ist gut geeignet um mögliche Kandidaten (für die Länge des Musters) ausfindig zu machen. Um die Auswahl zu verfeinern werden die Kandidaten in einem zweiten Schritt mit Hilfe der zyklischen Autokorrelation überprüft und fehlerhafte Kandidaten verworfen. Aus den verbleibenden Musterlängen wird die mit größtem Autokorrelationswert ausgewählt.

Da die zugrundeliegenden Performance-Daten allerdings bereits Anomalien enthalten können, wird die Zeitreihe in einem zusätzlichen Vorverarbeitungsschritt so aufbereitet, dass der Einfluss von möglichen Anomalien reduziert wird. Dazu werden statistische Ausreißer auf den 1,5-fachen Interquartilsabstand limitiert, um, für den Fall dass sie Teil des Musters sind, immer noch Einfluss nehmen zu können und das Muster aber nicht durch einen einzigen Ausreißer verfälscht wird. In einem weiteren Schritt wird der Trend der Zeitreihe entfernt, da auch dieser das *autoperiod*-Ergebnis stark verfälschen kann. Dazu wird durch die Seasonal-Trend Dekomposition basierend auf Loess (STL) [CCMT90] der additive Trend bestimmt und anschließend von der Zeitreihe abgezogen.

Sobald die Länge des Musters bekannt ist bzw. geschätzt werden kann, lässt sich für die komplette Zeitreihe ein „durchschnittliches“ Muster berechnen. Dazu werden alle Einträge der Zeitreihe, die demselben Zeitpunkt im Muster entsprechen, zu einem Repräsentanten zusam-

mengefasst. Um auch hier den Einfluss von Ausreißern zu reduzieren, wird der Median anstatt des Durchschnitts verwendet.

Wird ein neuer Messwert von *Icinga* empfangen, kann er mit dem entsprechenden Eintrag im Muster verglichen werden. Basierend auf den bisherigen Vorhersagefehlern wird eine gewisse Toleranz um den Vorhersagewert eingeräumt, in dem der Messwert liegen darf, ohne als Anomalie zu gelten. Zusätzlich ist auch eine leichte Verschiebung in der Zeit erlaubt, da Muster über Tage oder Wochen hinweg nicht unbedingt immer auf exakt demselben Zeitpunkt liegen.

Da einzelne Anomalien auch durch Rauschen auftreten können, werden Alarmmeldungen an *Icinga* nur ausgegeben, wenn eine bestimmte Anzahl an Anomalien über einen gewissen Zeitraum auftreten. Um dennoch alle gefunden Informationen in *Icinga* sammeln zu können, werden einzelne Anomalien als Warnung gemeldet.

4 Ereignis-Korrelation

Um einen möglichst umfangreichen Nutzen aus *Icinga*-Sensorereignissen ziehen zu können, wurde *Icinga* um eine Ereigniskorrelation ergänzt. Diese ermöglicht, Ereignisse sowohl mit Hintergrundwissen als auch Ereignisse untereinander in Beziehung zu setzen bzw. zu korrelieren. Unter *Hintergrundwissen* sind zum Beispiel Informationen über IT-Assets, Prozessschritte mit deren Abhängigkeiten oder eingesetzte Software und Softwareversionen mit deren Verwundbarkeiten zu verstehen. So kann beispielsweise geprüft werden, ob bei einem Ausfall eines Hosts wichtige betriebliche Prozesse gefährdet sind oder ob eine erkannte Anomalie im Zusammenhang mit einer Verwundbarkeit steht.

In *iMonitor* wurde *Icinga* um eine solche Ereigniskorrelation basierend auf einer sogenannten *Ontologie* bereichert. Eine *Ontologie* ermöglicht die Repräsentation von SIEM relevanten Informationen in einer strukturierten Form. Diese Art der Repräsentation wurde bereits mehrfach für die Modellierung von sicherheitsrelevanten Informationen verwendet, wie z.B. in [GMHD12] oder [OCM12]. zeigt einen Ausschnitt der Ontologie, die alle statischen Informationen in Konzepte strukturiert, wie z.B. das der kritische Servicezustand (*ServicestateCritical*) ein besonderer Servicezustand (*Servicestate*) ist. Diese allgemeine Formulierung wird in einer Ontologie Konzept genannt. Ein Individuum ist dabei eine Instanz dieses Konzepts, z.B. das Individuum *servicestateCritical* das den konkreten, möglichen Zustand in einer Umgebung beschreibt. Ein Individuum kann dabei weitere konkrete Ausprägungen besitzt, z.B. durch ein *data property* kann beschrieben werden, dass der konkrete Servicezustand, wie in der Abbildung zu sehen, in der Einsatzumgebung den Namen *ServicestateCritical* oder *CRITICAL* besitzt. Diese konkreten Namen entsprechen dabei den Werten aus dem zu korrelierendem Ereignis, z.B. sendet *Icinga* ein Ereignis mit dem Wert *CRITICAL* für einen Servicezustand. Über die Ontologie kann festgestellt werden, dass dieser Zustand zu dem Konzept *ServicestateCritical* gehört. Der Vorteil dieser Information ist, dass Korrelationsregeln dadurch unabhängig von den konkreten Einsatzumgebungen erstellt werden können, indem nur das allgemeine Konzeptwissen verwendet wird und kein individuelles Wissen.

Die *iMonitor* Ereigniskorrelation wurde als ein von *Icinga* grundsätzlich unabhängiges Modul entwickelt. Über die in *Icinga* verfügbaren Ereignis-Verarbeiter (engl. event handler) kann dieses externe Modul jedoch ohne großen Aufwand zur Ereignisverarbeitung angesteuert

werden. Zum Datenaustausch wurde das *JSON*-Format verwendet, das es ermöglicht, menschenlesbare strukturierte Daten auszutauschen.

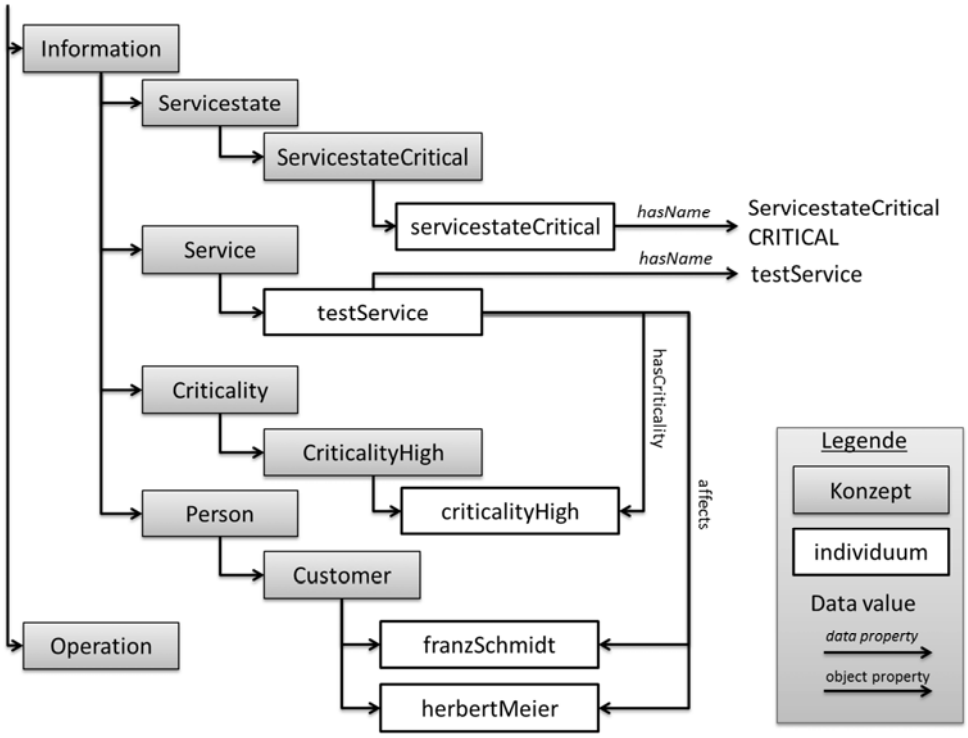

Abb. 3: Auszug aus der Ontologie

Die *JSON* Ereignisdaten werden von dem Korrelationsmodul temporär in die Ontologie übertragen. Somit sind in der Ontologie alle für die Ereigniskorrelation relevanten Informationen verfügbar. In der Ontologie wird sowohl eher statisches Hintergrundwissen, z.B. über IT-Assets, Services oder Kunden hinterlegt, als auch dynamische Informationen aus dem zu verarbeitendem Ereignis wie z.B. die Verfügbarkeit eines Hosts.

Über die Zusammenführung der Informationen in eine Ontologie können bereits etablierte Schlussfolgerungsalgorithmen wie z.B. aus *Pellet* [C15] als auch Anfragesprachen wie *SPARQL* [W15] für die Ereignis-Korrelationen verwendet werden. Mögliche Szenarien sind hier die Prüfung, welche Dienste von einer bestimmten IT-Komponente abhängig sind oder welche möglichen Kunden von einem Ausfall einer Komponente betroffen sind. In beschreibt beispielsweise die Relation (engl.: object property) *affects*, dass die Kunden *franzSchmidt* und *herbertMeier* von dem Ausfall des Services *testService* betroffen sind.

Da eine Ontologie flexibel bezüglich der Art des modellierbaren Wissens ist, da nahezu beliebig Relationen, Konzepte und Individuen erstellt werden können, können auch komplexere Anfragen vorgenommen werden, zum Beispiel: *Gibt es bekannte Verwundbarkeiten der Software auf dem Host bei dem ein Sensor gerade ein auffälliges Verhalten (eine Anomalie) erkannt hat*.

Die in *iMonitor* verwendete Ontologie wurde analog zu der von Granadillo et al. vorgestellten Ontologie für SIEM-Systeme erstellt [GMHD12]. Dabei wird diese Wissensstruktur, wie in zu sehen, grundsätzlich in die zwei Bereiche *Operation* und *Information* unterteilt. In *iMonitor* werden unter der Klasse *Information* alle statischen Hintergrundinformationen gesammelt, die für alle Ereignisse zur Verfügung stehen, wie z.B. IT-Assets oder Prozesse. Die Klasse

Operation wird für die Korrelation zur Laufzeit verwendet. Hier hat der Anwender die Möglichkeit, neue Individuen anhand der eingehenden Ereignisse zu erstellen und diese auch für die spätere Korrelation wieder zu verwenden. Dies ermöglicht über eine in *iMonitor* umgesetzte Ergänzung der *SPARQL*-Anfragesprache die Korrelation der Ereignisse mit dem Hintergrundwissen als auch eine zeitliche Korrelation der Ereignisse, also die Korrelation der Ereignisse zu bereits vorher eingetretenen Ereignissen. Dies wurde über einfache Ersetzungen in der Anfrage realisiert, bei dem bestimmte Makros bzw. Platzhalter durch Werte des Ereignisses ersetzt werden. Dabei ist jeder Variablenwert aus den *JSON*-Daten als auch zusätzliche Werte wie z.B. die aktuelle Zeit in einer Anfrage an die Ontologie verfügbar. Zur einfachen Verwendung in den Anfragen sind die *JSON*-Daten auf oberste Ebene direkt über Makros verfügbar. In der folgenden *SPARQL*-Anfrage wird zum Beispiel geprüft, ob der Ereigniswert für die Ereignisvariable *SERVICESTATE* in der Ontologie als ein kritischer Servicezustand (*ServicestateCritical*) bekannt ist:

```
?servicestate imonitor:hasName “$SERVICESTATE“.
?servicestate a imonitor:ServicestateCritical.
```

Der Platzhalter *$SERVICESTATE* würde über die Korrelation von einem *Icinga*-Ereignis, das einen kritischen Servicezustand beschreibt, durch den Wert *CRITICAL* ersetzt werden. Der Ausdruck *?servicestate* beschreibt wie in SPARQL üblich eine Variable, deren Belegung in diesem Beispiel einem Individuum entsprechen muss, dass sowohl vom Typ ein *Servicestate-Critical* sein muss, als auch eine Relation *hasName* mit dem Wert *CRITICAL* besitzen muss.

Komplexere, verschachtelte Ereignis-Daten (z.B. eine Baum-Struktur aus *JSON*) stehen in der Ontologie als miteinander verknüpfte Individuen zur Verfügung.

Jede Korrelationsregel besitzt nicht nur einen Bedingungsteil, bei dem bestimmte Eigenschaften aus der Ontologie im Zusammenhang mit dem Ereignis überprüft werden können, sondern auch über einen Aktionsteil, bei dem z.B. spezifiziert werden kann, ob neue Elemente der Ontologie hinzugefügt werden sollen, um diese für eine spätere Korrelation zu verwenden, oder ob ein Vorfall vorliegt, der z.B. über ein Ticketsystem eskaliert werden muss.

Zudem kann im Aktionsteil einer Regel auch eine Erklärung des Vorfalls wie auch eine Handlungsempfehlung generiert werden. Dazu sammeln *SPARQL*-Anfragen Hintergrundinformationen, die für eine Handlungsempfehlung oder Vorfallerklärung benötigt werden. Diese können wiederum in natürlich-sprachlichen Texten referenziert werden. Beispielsweise kann die Liste der betroffenen Kunden in einer Variable *$customers* zwischengespeichert werden, um dann in der Handlungsempfehlung wie z.B. *Es sind die Kunden $customers über den Vorfall zu informieren.* über ein Makro referenziert zu werden.

5 Erkennung von Vorfallvariationen

Um auch Variationen von Vorfällen erkennen zu können, wurde die SPARQL Anfragesprache zudem durch Abstraktionsfunktionen erweitert. Mit einer Abstraktionsfunktion können Teile einer *SPARQL*-Bedingungen gekennzeichnet werden. Diese Kennzeichnung führt dazu, dass für den Fall, dass keine Regel auf ein Ereignis greift, diese Bedingung automatisch von der Korrelation abstrahiert werden darf. Geht man beispielsweise davon aus, dass ein Webserver als ein Unterkonzept eines Servers definiert ist, wäre die Abstraktion der Bedingung „ist ein Webserver“ die Bedingung „ist ein Server“. Eine Ontologie bietet für die Abstraktion der Bedingungen bereits die relevanten Informationen. Ein Verfahren zur Abstraktion mit Hilfe

einer Ontologie wurde bereits in [HCYP04] vorgestellt. Anders als in der genannten Arbeit, die ein schwellwertbasiertes System vorschlägt, kann in *iMonitor* je nach Bedingungsteil direkt in einer *SPARQL*-Anfrage der Grad der maximalen Abstraktion individuell festgelegt werden.

Die Abstraktion wird solange wiederholt, bis mindestens eine Regel das Ereignis behandelt oder das an der Abstraktionsfunktion anzugebende Abstraktionsmaximum erreicht ist. Die maximale Abstraktion erlaubt dem Anwender einen Vorfall auch dann einzustufen, wenn keine Regel direkt für diesen Vorfall existiert. Dabei ist zu berücksichtigen, dass die Regel aufgrund der Abstraktion nicht direkt greift, aber auch die Information, welche ähnliche Regel vorhanden ist, kann für den Anwender hilfreich sein, um den Vorfall bewerten zu können, z.B. das eine entsprechende Regel für einen Webserver existiert, das Ereignis sich jedoch auf eine andere Serverart bezieht. Um auch Handlungsempfehlungen und Erklärungen automatisch an eine Abstraktion anzupassen, kann wieder über eine in *SPARQL* zu verwendende Funktion die Abstraktion einer Regel abfragen, um somit auch die Hintergrundinformationen zum Vorfall automatisch zu abstrahieren.

6 Wissensaustausch

Um den Modellierungsaufwand in einem Unternehmen möglichst gering zu halten, wurde in *iMonitor* eine Methode entwickelt, die es ermöglicht, dass Unternehmen ihre Korrelationsregeln auch anderen Unternehmen zur Verfügung stellen. Darüber hinaus gibt es die Möglichkeit, vorgefertigte Korrelationsregeln ohne Integrationsaufwand von einem zentralen Server, dem Wissensserver wie in **Abb. 4** zu sehen, herunter zu laden und zu verwenden. Um dies zu ermöglichen, wurde die natürliche Trennung einer Ontologie in *T-Box* und *A-Box* genutzt (s. [BESS11]). Im T-Box-Teil werden lediglich allgemeine Informationen beschrieben (die Konzepte in), wie z.B. die Kritikalität oder grundlegende Konzepte über Workstations, Server oder IT- und Business-Prozesse. Über dieses Wissen verfügen alle *iMonitor-Clients* und können somit direkt Regeln verwenden, die nur solches, allgemeines Wissen (aus der T-Box) benötigen. Zum Beispiel kann eine Regel besagen, dass beim Ausfall eines (beliebigen) IT-Prozesses der für einen (beliebigen) Business-Prozess mit hoher Kritikalität entscheidend ist, ein Vorfall erstellt werden muss.

Bevor eine Regel von einem Unternehmen zur allgemeinen Verfügung gestellt wird, wird vom *iMonitor-Cient* geprüft (s. **Abb. 4**), ob kein individuelles (z.B. unternehmensspezifisches) Wissen verwendet wurde. Dabei wird überprüft, ob ausschließlich Elemente aus der T-Box der Ontologie verwendet wurden um die Regelbedingungen, die Erklärungen und die Handlungsempfehlungen zu modellieren. Ist dies nicht der Fall, kann die Regel nicht ausgetauscht werden, da diese zum einen unternehmensspezifische Informationen enthalten kann, die nicht geteilt werden sollten, aber auch da die Regel nicht ohne manuelle Anpassungen in eine andere, spezifischere Umgebung integriert werden kann. Um dies von vornherein auszuschließen, wird zusätzlich von dem Wissensserver nochmals geprüft, dass kein individuelles Wissen bzgl. der aktuellen, gemeinsamen „top-level“ Ontologie verwendet wurde. Dies gewährleistet, dass ein Missbrauch, z.B. mit manipulierten Clients die individuelle Regeln senden, bereits vor der manuellen Qualitätssicherung, automatisch verhindert werden kann.

Problematisch bleibt die automatische Überprüfung der Regel bezüglich der Semantik. Um zu verhindern, dass unsinnige Regeln ausgetauscht werden, die nicht dem erwarteten Verhalten entsprechen, findet ein manueller Qualitätssicherungsprozess statt. Der zentrale Server schal-

tet eine ihm zugesandte Regel nicht direkt zum Download für andere *iMonitor Clients* frei, sondern jede Regel erfordert eine manuelle Prüfung.

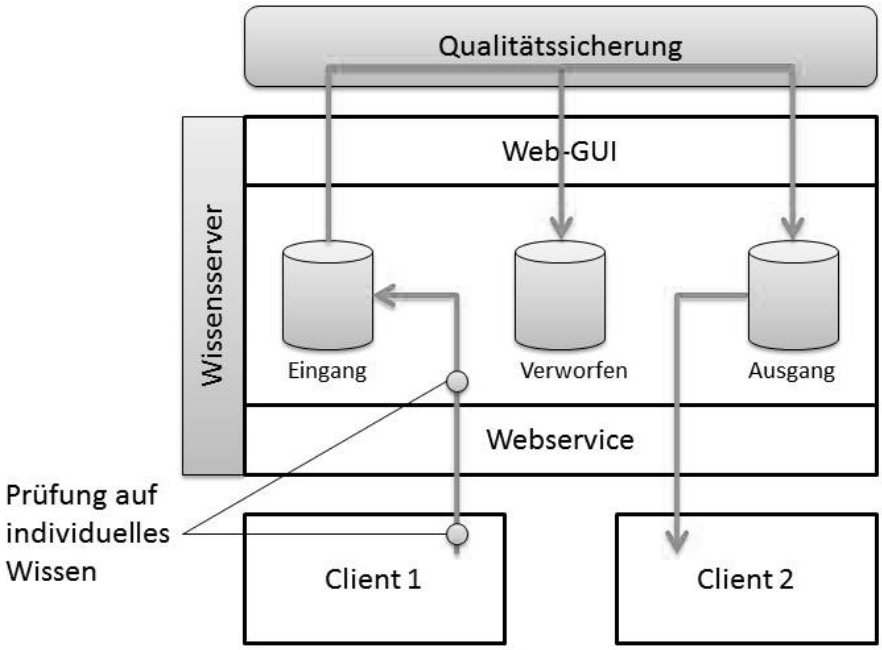

Abb. 4: Wissensaustausch

7 Anwendungsfall

Ein Beispielanwendungsfall für die entwickelten Komponenten wäre im ersten Schritt die Erhebung der aktuellen Infrastruktur mithilfe des IO-Tool-Sets. Dieses generiert aus den erhobenen Daten dann die ersten *Icinga*-Konfigurationsdateien. Ist das IO-Tool-Set nicht integriert, können auch bereits existierende Konfigurationsdateien aus einer bereits bestehenden *Nagios/Icinga* Instanz verwendet werden oder müssen neu erstellt werden.

Der Ablauf in *iMonitor* kann in drei unterschiedliche Schritte unterteilt werden (siehe **Abb. 5**). Der erste ist die Datenerhebung, in dem die Sensor-Daten (Netzverkehr, Schwachstellen-Scan-Daten, Performance-Daten, usw.) gesammelt werden. Die *iMonitor*-Kollektoren greifen auf die Logs zu und senden diese per NSCA, NRPE oder einem Plug-In an *Icinga*, das diese dann in seiner Datenbank einfügt.

Die Zeitreihenanalyse ist sowohl in der Datenerhebung als auch in der Datenverarbeitung angesiedelt. Sie greift auf die *Icinga* Datenbank zu und analysiert die Performance-Daten nach Anomalien (Abweichungen zum Normalbetrieb). Entdeckt die Zeitreihenanalyse eine Anomalie, wird diese ebenfalls an *Icinga* gesendet (Datenerhebung). Hierbei kann es direkt nach der Inbetriebnahme zu einer Häufung von Fehlalarmen kommen (siehe Zeitreihenbasierte Anomalie-Erkennung). *Icinga* selbst ist wie die Zeitreihenanalyse ebenfalls in den Schritten Datenerhebung und Datenverarbeitung angesiedelt. Es werden die Monitoring-Daten der eingebundenen Systeme erfasst und verarbeitet.

Die *Korrelation* verarbeitet die gespeicherten Daten aus *Icinga* und wertet die einzelnen Ereignisdaten anhand von definierten Regeln aus. Trifft eine Regel zu (match), so erstellt die

Korrelation einen Eintrag in der *Vorfallsdatenbank* mit entsprechenden Handlungsempfehlungen und Erklärungen bezüglich des Vorfalls. Des Weiteren stuft die *Korrelation* die Vorfälle bezüglich eines in den Regeln definierten Risikowertes ein.

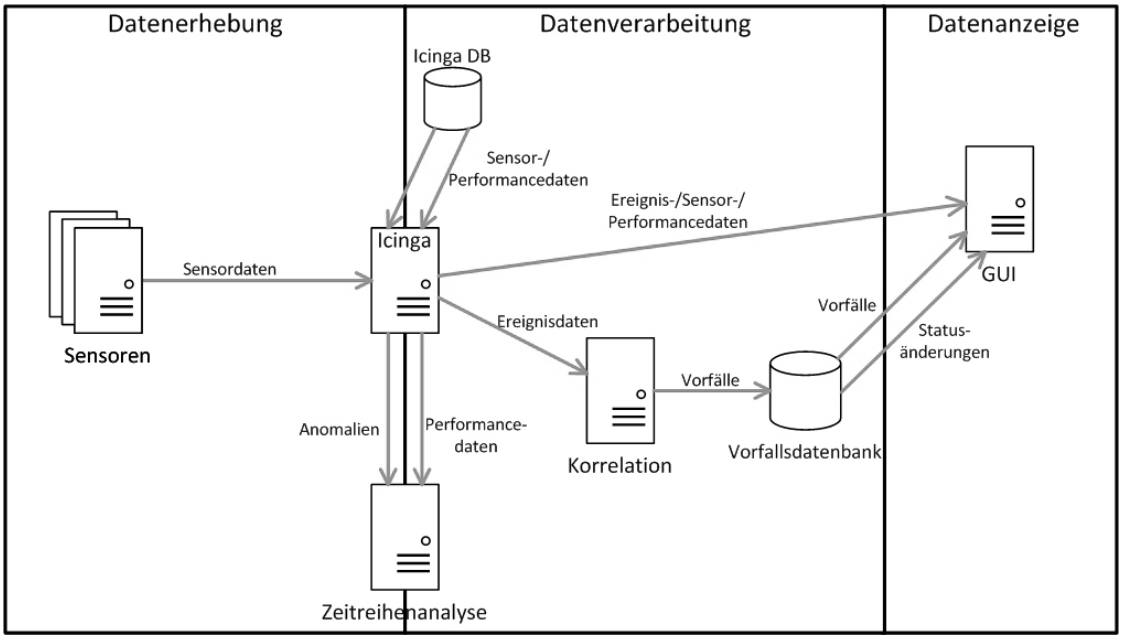

Abb. 5: Datenfluss im *iMonitor*

Die *GUI* greift sowohl auf die *Icinga*-Datenbank zu wie auch auf die *Vorfallsdatenbank* und zeigt die Events an. Wird ein neuer Vorfall in der Datenbank erkannt so legt die *GUI* auch ein entsprechendes Ticket in dem angeschlossenem Ticketsystem an. Weiterhin zeigt die *GUI* den aktuellen Bedrohungsstatus der überwachten Umgebung an. Die erstellten Tickets können in der GUI eingesehen und bearbeitet werden. Dabei ist es möglich den aktuellen Bearbeitungsstatus zu den Vorfällen abzufragen. Zusätzlich können Kommentare angelegt werden, wie z.B. der Vorfall behoben wurde.

Des Weiteren unterstützt die *Korrelation* einen Wissensaustausch über einen zentralen Server mit anderen *iMonitor* Komponenten bei dem gewährleistet wird, dass keine Unternehmensspezifischen Informationen übertragen werden.

8 Zusammenfassung

Heutige SIEM-Systeme arbeiten bisher ähnlich, wie Anti-Viren- oder Anti-Spam-Programme. Sie erkennen anhand bekannter Muster, ob eine Anomalie vorliegt oder nicht. Ist das Muster nicht bekannt, kann sich der Angriff auch weiterhin ungehindert ausbreiten. Diesen Ansatz versucht *iMonitor* zu umgehen, indem KI-Ansätze zur Zeitreihenanalyse mit eingebracht werden, die zum einen eine Anomalie erkennen sollen, bevor die Mustererkennung anschlägt und zum anderen das Normalverhalten des Netzes kennenlernen bzw. erfassen. Erst wenn das Normalverhalten bekannt ist, kann ein anomales Verhalten erkannt und analysiert werden. Dabei baut *iMonitor* auf der bekannten Monitoring-Lösung *Icinga* (*Nagios*) auf, die bereits sehr umfangreiche Mechanismen zur Verfügbarkeitsüberwachung besitzt, und erweitert sie um das Monitoring der IT-Sicherheit. Über die SIEM-GUI lassen sich dann Handlungsempfehlungen an die IT-Administration weiterleiten, um effektiv und schnell auf Gefahrenquellen hinweisen zu können.

9 Danksagung

Das *iMonitor*-Projekt (www.imonitor-project.de) ist ein gefördertes BMWi-Projekt mit einer Laufzeit von zwei Jahren, das im Juli 2013 seine Arbeiten begonnen hat. An dem Projekt sind die Firmen *DECOIT GmbH* (Projektleitung, Entwicklung) und *neusta software development* (Entwicklung) sowie das *Technologie-Zentrum für Informatik und Informationstechnik (TZI)* der Universität Bremen beteiligt. Daher gilt der Dank den Partnern des Projektes, die durch ihre Beiträge und Arbeiten diesen Bericht erst ermöglicht haben.

Literatur

[BESS11] Henk Birkholz, Carsten Elfers, Bernd Samjeske und Karsten Sohr, "Unternehmensbergreifender Austausch von sicherheitsrelevantem Wissen" Datenschutz und Datensicherheit (DuD), vol. 4, pp. 258-261, 2011

[C15] Clarkspasia- Pellet, Website, http://clarkparsia.com/pellet/, 2015

[CCMT90] R. B. Cleveland, W. S. Cleveland, J. E. McRae und I. Terpenning: STL: A seasonal-trend decomposition procedure based on loess. In: *Journal of Official Statistics*, Vol. 6, Nr. 1, S. 3-73, 1990

[DRS14] Kai-Oliver Detken, Rossow und Steuerwald: *SIEM-Ansätze zur Erhöhung der IT-Sicherheit auf Basis von IF-MAP*. D.A.CH Security 2014: Bestandsaufnahme, Konzepte, Anwendungen und Perspektiven, ISBN 978-3-00-046463-8, Hrsg. Peter Schartner u. Peter Lipp, syssec-Verlag, Graz (Österreich) 2014

[GMHD12] Gustavo Gonzalez Granadillo, Yosra Ben Mustapha, Nabil Hachem und Herve Debar. An ontology-driven approach to model SIEM information and operations using the SWRL formalism. Int. J. of Electronic Security and Digital Forensics. 2012, Vol. 4, S. 104-123

[HCYP04] Y. He, W. Chen, M. Yang und W. Peng. Ontology based cooperative intrusion detection system. In Network and Parallel Computing, Lecture Notes in Computer Science, Springer Berlin, 2004, S. 419-426.

[OCM12] Obrst, L., Chase, P., Markeloff, R. Developing an Ontology of the Cyber Security Domain. Seventh International Conference on Semantic Technologies for Intelligence, Defense, and Security: 49-56

[W15] W3C - SPARQL, Website, http://www.w3.org/TR/rdf-sparql-query/, 2015

[VYC05] Michail Vlachos, Philip Yu, Vittorio Castelli. *On Periodicity Detection and Structural Periodic Similarity*. Proceedings of the 2005 SIAM International Conference on Data Mining, 2005, S. 449-460