TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Chair of Privacy and Data Security

Bachelor-Arbeit

Kontextsensitivität für Netzwerk-Sicherheits-Monitoring

Tobias Köllner

Geboren am: 25. Dezember 1999 in Dresden
Matrikelnummer: 4803900
Immatrikulationsjahr: 2018

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc)

Betreuer
M.Sc. Sebastian Rehms

Betreuender Hochschullehrer
Dr. Stefan Köpsell

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik • Institut für Systemarchitektur • Professur für Datenschutz und Datensicherheit

AUFGABENSTELLUNG FÜR DIE BACHELORARBEIT

Name, Vorname des Studenten: Köllner, Tobias
Immatrikulationsnummer: 4803900
Studiengang: Bachelor Informatik
E-Mail: tobias.koellner@mailbox.tu-dresden.de

Thema: Kontextsensitivität für Netzwerk-Sicherheits-Monitoring

Zielstellung:

Zur Erhöhung der Sicherheit von Netzwerken werden häufig Techniken wie Network security monitoring (NSM) sowie intrusion detection (IDS) verwendet. Inbsesondere die Menge an falsch-positiven Meldungen stellt hierbei eine große Herausforderung dar. Um dieses Problem sowie die generelle Interpretierbarkeit der Beobachtungsdaten zu verbessern können zusätzliche Informationen gesammelt werden, die man als Netzwerkkontext bezeichnen kann.

Der erste Teil der Arbeit soll untersuchen, wie solche Kontextinformationen sowohl während des Betriebs von NSM-Systemen als auch während der Signatur/Modell-Erstellung im Rahmen von IDS gesammelt und verwendet werden können. Ziele hierbei sind, solche Möglichkeiten auf theoretischer Ebene wie auch von open-source Implemntierungen zu sichten und hierbei einen kritischen Kontextbegriff im Vergleich zu etablierten Kontextbegriffen in der Literatur in Bezug auf Netzwerke zu entwickeln.

Aufbauend auf diesen Resultaten kann eine einfache praktische Auswertung verfügbarer Implementierungen mit explizitem Hinblick auf die kontextsensitiven Fähigkeiten durchgeführt werden. Hierfür ist ein Plan für eine minimale Testumgebung sowie das Design von geignetem Netzwerkverkehr erforderlich.

In der Arbeit sollen schwerpunktmäßig folgende Teilaufgaben bearbeitet werden:

- Überblick und Analyse existierender NSM-Ansätze in Bezug auf Kontext- und Umgebungsbewusstsein. Hierbei sollen mindestens folgende Forschungsfragen bearbeitet werden:
  - Lassen sich generelle Kategorien von Kontext im Rahmen von NSM formulieren?
  - Welche Kontextinformation kann während der Regelgenerierung im Unterschied zu generell verfügbaren Regel aus öffentlichen Quellen berücksichtigt werden?
  - Welche Techniken wurden vorgeschlagen, um Kontextinformation während des regulären Betriebs zu sammeln und wie werden diese dann genutzt?
  - Inwifern gibt es Möglichkeiten der Anpassungen an Änderungen der Netzwerkumgebung, etwa der Topologie oder der genutzen Protokolle?
  - Wie stark trägt Kontextinformation zur Reduktion von falsch-Positiven bei?
- Es soll eine Evaluation der Fähigkeit von Kontextnutzung von verfügbaren open-source-Implementierungen (etwa Plugins in Zeek oder Netzdefinitionen in Suricata) durchgeführt werden.

Verantwortlicher Hochschullehrer: Dr. Stefan Köpsell
Institut: Systemarchitektur
Beginn am: 3.6.2022 Einzureichen am: 19.8. 2022

3.6.22, T. Köllner .....
Datum, Unterschrift der/des Studierenden

.....
Unterschrift des betreuenden Hochschullehrers

Kopie: Sekretariat + elektronische Datei

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel *Kontextsensitivität für Netzwerk-Sicherheits-Monitoring* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in der Arbeit angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Während der Anfertigung dieser Arbeit wurde ich nur von folgenden Personen unterstützt:

M.Sc. Sebastian Rehms

Weitere Personen waren an der geistigen Herstellung der vorliegenden Arbeit nicht beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Tobias Köllner

Die Aufrechterhaltung und Verbesserung der Netzwerksicherheit, sowie die Anpassungsfähigkeit der dazu eingesetzten Schutzmechanismen an immer neue Gegbenheiten, sind seit dem bestehen von IT-Netzwerken von ein ständiges Problem. Netzwerke werden immer weitreichender und komplexer. Sowohl die in das Netzwerk zu integrierenden und zu verwaltenden Systeme, als auch deren Verwendungsmöglichkeiten werden immer vielfältiger. Dadurch unterliegen auch das damit verbundene immense Netzwerkverkehrsaufkommen und die dadurch möglich werdenden Angriffe einem stetigen Wandel. Deshalb müssen auch bisher erdachte Schutzkonzepte immer wieder angepasst und die dafür verwendete Software stetig verbessert werden. Eins der zentralen Konzepte um diese Aufgaben zu bewältigen ist Kontextsensitivität. Diese Arbeit gibt einen Überblick über den aktuellen Stand im Bereich der Kategorisierung von Kontext. Präsentiert eine Variante diese unter dem Gesichtpunkt des Netzwerk-Sicherheits-Monitoring in Verbindung zu bringen. Zeigt auf wo und wie diese Informationen gesammelt werden können. Betrachtet werden dabei konkrete Leistungskriterien mit Hilfe ausgewählter Anwendungsfälle. Darauf aufbauend erfolgt abschließend eine Bewertung der vorgestellten Kontextkategorien im Bezug auf den Nutzen zur Verbesserung eines IDS. Das soll den Lesenden dazu befähigen, selbst Anpassungen an der Arbeitsweise des eigenen Systems vorzunehmen und so die Netzwerksicherheit nachhaltig zu erhöhen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

1 Einführung

2 Hintergrund

2.1 Kontextbegriff

Kontext ist zwar ein Begriff, unter dem sich die meisten Menschen intuitiv etwas vorstellen können, ihn zu erläutern oder gar korrekt und vollständig zu definieren, fällt allerdings um ein Vielfaches schwerer [8]. Auch herrschen in verschiedenen Fachbereichen jeder davon mit anderen Sachverhalten und Problemen, welche die jeweiligen Autoren versuchen zu lösen, abweichende Auffassungen darüber, welche Ansprüche eine Definition erfüllen muss. Das erschwert eine allumfassende, konkrete Bestimmung zusätzlich [5]. Ziel der Arbeit ist also nicht, Kontext abschließend zu definieren. Dies ist in Anbetracht der vielen verschiedenen Ansätze und dem fehlenden Konsens in der Literatur darüber, wie solch eine Definition aussehen soll [4, 24] nicht zufriedenstellend möglich. Die Menge an Anwendungsfällen und damit auch die zu beachtenden Gesichtspunkte, die man für eine allen Ansprüchen genügende Definition benötigt, sind dafür zu umfangreich. Stattdessen wird versucht, mithilfe relevanter Darlegungen anderer Autoren die historische Entwicklung des Kontextbegriffs für den Bereich der Informatik im Allgemeinen und der Zugriffskontrolle im Speziellen darzustellen. Dies soll ein Verständnis dafür schaffen, auf welchen Grundlagen, Ansätzen und Ideen der Kontextbegriff dieser Arbeit entstanden und aufgebaut ist. Dies ermöglicht dem Leser eine Einordnung davon, wie Kontext und Kontextsensitivität verwendet werden und er kann einen Abgleich mit seiner eigenen Interpretation der Begrifflichkeiten durchführen.

2.1.1 Historische Entwicklung

Die Definition von Kontext ist auch in der Literatur seit jeher ein Thema. Eine der frühesten von Schilit et al. [21] bestimmt als drei Hauptaspekte, an welchem Ort, in Gegenwart welcher anderen Personen und in der Nähe welcher Ressourcen sich ein Nutzer befindet. Des Weiteren beinhaltet nach [21] Kontext Attribute wie Beleuchtung, Lautstärke, den Grad der Netzwerkverbindung, Kommunikationskosten, und die soziale Situation. Eine der seit ihrer Entstehung in 2001 am häufigsten zitierten Definitionen stammt von Dey [8]. Laut dieser ist Kontext "jede Information, die genutzt werden kann, um die Situation einer Entität zu charakterisieren. Eine Entität ist eine Person, ein Objekt oder ein Ort mit Relevanz für die Interaktion zwischen Nutzer und Anwendung. Das schließt auch Nutzer und Anwendung selbst mit ein". Sie wird allgemein hin von den vielen anderen Autoren als Quasikonsens akzeptiert [3, 4, 24] oder sogar als Ausgangspunkt für ihre eigene Definition genutzt [13, 27]. Kaltz et al. [25] verstehen Kontext als einen Kontextraum, also eine Kombination aus Kontextparametern, Elementen einer domänenspezifischen Ontologie und Dienstleistungsbeschreibungen in Form von $C = \{U; P; L; T; D; I; S\}$. Dabei ist U das Set aus Nutzern und den dazugehörigen

Rollen, *P* die Prozesse und Aufgaben, *L* der Ort, *T* der Zeitfaktor, *D* beschreibt das Gerät, *I* die verfügbaren Informationen und *S* die verfügbaren Dienstleistungen. Ein spezifischer Kontext ist somit ein Punkt in diesem Raum. Einen anderen Ansatz wählen Bazire und Brézillon. Sie haben 150 Kontextdefinitionen analysiert und sind dabei zu der Erkenntnis gekommen, das Kontext wie eine Begrenzung fungiert, welche das Verhalten eines Systems, Nutzers oder Computers in einer bestimmten Tätigkeit beeinflussen. Allerdings herrscht ihrer Ansicht nach kein Konsens darüber, ob Kontext extern oder intern, ein Set aus Informationen oder Abläufen, statisch oder dynamisch ist [5]. Im Bezug auf kontextsensitive Zugriffskontrolle sind die Arbeiten von Kayes et al. [13, 14, 15] erwähnenswert. In [13] definieren diese Kontextinformationen, in Bezug auf Zugriffskontrollentscheidungen, als relevante Informationen über den Zustand einer Entität (Nutzer, Ressource, Ressourcenbesitzer) und deren Umgebung oder die Beziehung zwischen Entitäten.

2.2 Kontextkategorisierung

Genauso relevant für diese Arbeit wie eine Übersicht gängiger Definitionen ist ein Überblick darüber, wie Kontext kategorisiert werden kann. Nach dem Kontextverständnis in der Literatur wird also im Folgenden die Kategorisierung von Kontext näher betrachtet. Dazu erfolgt in diesem Kapitel eine Vorstellung verschiedener ausgewählter Unterteilungsansätze. Auf diese wird sich auch bei der Erstellung der Taxonomie dieser Arbeit in 4 bezogen.

2.2.1 Übersicht bereits vorgeschlagener Kategorien

Auch bei der Kategorisierung von Kontext gibt es richtungsweisende Vorschläge. Abowd et al. [2] haben einen der wegweisenden Mechanismen zur Definition von Typen von Kontext vorgeschlagen. Sie identifizierten Ort, Zeit, Identität und Aktivität als primäre Kontexttypen. Weiterhin wird sekundärer Kontext als etwas definiert, das durch Nutzung von Primärkontext erschlossen werden kann. Schilit et al. [21] kategorisieren Kontext basierend auf drei Fragen, die genutzt werden können, um den Kontext zu bestimmen, in drei Kategorien:

1. Informationen, die sich auf einen Ort beziehen, beispielsweise GPS-Koordinaten, Bezeichnungen von Institutionen oder Gebäuden (ein Café, ein Krankenhaus, eine Universität), spezifische Namen (z. B.: Technische Universität Dresden) spezifischen Adressen(z. B.: APB Nöthnitzer Str. 46) oder Nutzerpräferenzen (z. B. das Lieblingsrestaurant eines Nutzers)

2. Informationen über andere Personen, die in der Nähe aufhalten

3. Informationen darüber, welche Ressourcen (Maschinen, technische Geräte, Betriebsmittel) sich im direkten Umfeld eines Nutzers befinden

Henricksen [11] ordnet Kontext basierend auf der betrieblichen Kategorisierungstechnik in 4 verschiedene Kategorien:

1. Messbar: Informationen, die aus unmittelbar messbaren Werten bestehen. Diese ändern sich oft oder gar kontinuierlich.

2. Statisch: Informationen, die sich während der Lebenszeit eines Systems gleich bleiben.

3. Profiliert: Informationen, die sich selten ändern.

4. Abgeleitet: Informationen, die unter Verwendung anderer Daten gewonnen wurden.

Van Bunningen et al. [23] ordnen Kategorisierungsversuche in zwei übergeordnete Gruppen: Betrieblich und Konzeptionell.

1. betriebliche Kategorisierung: Einordnung anhand dessen, wie der Kontext akquiriert, modelliert und behandelt wird.

2. konzeptionelle Kategorisierung: Einordnung anhand der Bedeutung des Kontextes und der konzeptionellen Beziehungen

Chong et al. [6] schlagen Historie als Kontextkategorie vor. Dabei wird die zeitliche Entwicklung einer bestimmten Messgröße als Kontext definiert. Das erlaubt die Festlegung eines Normalzustandes für dieser Größe. Das ermöglicht laut den Autoren unter Umständen eine Vorhersage darüber, welche Werte die Messgrößen zukünftig annehmen könnten.

3 Voraussetzungen und verwandte Arbeiten

3.1 Netzwerk-Sicherheits-Monitoring

Ghafir et al. [9] beschreiben Sicherheits-Monitoring und dessen Aufgaben wie folgendermaßen. Bei Netzwerk-Sicherheits-Monitoring handelt es sich um eine Reihe von Mechanismen, die ermöglichen, den momentanen Zustand und langfristige Trends eines komplexen Computernetzwerks zu erkennen. Netzwerküberwachung umfasst mehrere Methoden, die gezielt eingesetzt werden, um die Sicherheit und Zuverlässigkeit aufrechtzuerhalten. Dabei reicht das Aufgabenfeld von Hardware, Software, Viren, Spyware und Schwachstellen wie Hintertüren bis zu Sicherheitslücken sowie anderen Aspekten, die die Integrität eines Netzwerks gefährden können. Um proaktiv statt reaktiv vorgehen zu können, muss der Datenverkehr und die Leistung im gesamten Netzwerk überwacht und sichergestellt werden, dass keine Sicherheitslücken im Netzwerk auftreten. Im Falle eines Netzwerkfehlers müssen Fehlfunktionen erkannt, isoliert und behoben werden. Bei einem stabilen Netz ist ständige Überwachung, ob eine Bedrohung von innerhalb oder außerhalb vorliegt, notwendig.

3.1.1 Intrusion Detection

Scarfone et al. [20] erklären Angriffserkennung, als "Prozess der Überwachung von Ereignissen in einem Computersystem oder Netzwerk und die Analyse dieser Ereignisse auf Anzeichen eines möglichen Zwischenfalls. Gemeint sind damit Verstöße oder unmittelbare Bedrohungen von Sicherheitsrichtlinien, Akzeptanzrichtlinien oder Standardsicherheitspraktiken".

3.1.2 Intrusion Detection System

Aufbauend auf der Definition von Intrusion Detection bezeichnen die Autoren ein IDS somit Software, die den Prozess, einen Eingriff zu erkennen, automatisiert [20]. Nach Jones et al. [12] werden Informationssysteme immer umfassender werden und haben einen stetig höher werdenden Wert für Netzwerksicherheit. Schneier [22] bezeichnet IDS vereinfacht als das Netzwerkäquivalent zu Virenscannern. IDS untersuchen den Netzwerkverkehr oder die auf den Hosts laufenden Prozesse auf Anzeichen für einen Angriff. Wenn sie einen solchen erkennen, schlagen sie Alarm. Seiner Ansicht nach sind IDS dabei nicht darauf ausgelegt, dass jeder Angriff identifiziert wird. Da es immer ein Angriffsszenario gibt, in dem ein ausreichend geschickter Angreifer ein Erkennungssystem umgehen können wird. Trotzdem stellt ein IDS als Teil der Detektionsphase von NSM eine wirksame Sicherheitsmaßnahme dar.

4 Design

In diesem Kapitel soll es darum gehen, welche Punkte bei einem Entwurf einer kontextsensitiven Lösung zur Leistungsverbesserung und Problembehebung eines IDS beachtet werden sollte. Dazu wird zuerst die Notwendigkeit einer Kontexttaxonomie erläutert. Dann erfolgt die Definition der Taxonomiekategorien. Danach wird festgelegt, in welcher Form die Kontextinformationen in den einzelnen Kategorien vorliegen müssen und wie man diese Informationen aus unterschiedlichen Quellen sammelt. Zusätzlich wird noch darauf eingegangen, welche Anforderungen ein IDS erfüllen sollte, welche davon im speziellen für diese Arbeit relevant sind und welche Schwächen des IDS mithilfe des später in diesem Abschnitt vorgeschlagenen Designs und Kontextsensitivität gelöst werden.

4.1 Notwendigkeit der Erstellung einer Taxonomie

Im Abschnitt 2 wurden viele verschiedene Schemata zur Kategorisierung vorgeschlagen, keines dieser Schemata ist aber ausreichend um den Ansprüchen dieser Arbeit gerecht zu werden. Auch eine Evaluation 16 verschiedener Arbeiten durch Perera et al. [18] legt nahe, dass keine Kategorisierung allein allen Anforderungen hinreichend gerecht werden kann. Deshalb wird also statt ein einzelnes Schema zu verwenden eine Kombination bereits vorgeschlagener Kategorisierungsschemata zum Ausgleich der Nachteile der einzelnen Bestandteile vorgeschlagen. Dazu werden zuerst die einzelnen Kategorien im Bezug auf kontextsensitive Zugriffskontrolle definiert. Die Kategorien müssen dafür abhängig davon, mit welchem Fokus sie der jeweilige Autor konstruiert hat, mehr oder weniger stark abgewandelt werden.

4.2 Erstellung einer Taxonomie

Die in dieser Arbeit präsentierte Taxonomie besteht aus 3 Komponenten. Einer Unterteilung mit Fokus auf menschliche Nutzer und Menschen generell, eine Kategorisierung aus dem Betrachtungswinkel eines Computernetzwerkes und einer Historie um die zeitliche Entwicklung der beiden Schemata einzuordnen und damit die Aussagekraft zu erhöhen. Die Abbildung 4.1 und Abbildung 4.2 [1] veranschaulichen dabei die Beziehung der Unterkategorien zueinander und sind jeweils Beispiele dafür versehen, welche Kontextinformationen in den einzelnen Kategorien vorkommen können.

[1] Als Inspiration für den Stil und die Strukur der Grafiken diente eine in [18] verwendete Abbildung. Diese wurde inhaltlich adaptiert, um für den Kontext der Netzwerksicherheit verwendet werden zu können.

4.2.1 Mensch

Eine naheliegende Sichtweise zur Kategorisierung von Kontext, welche auch von vielen Autoren, auf die im Kapitel 2 eingegangen wurde, verwendet wird, ist die aus der Perspektive des Nutzers, also einer Person. Auch für eine Einordnung im Rahmen der Zugriffskontrolle lässt sich argumentieren, dass ein Fokus auf Menschen sinnvoll ist. Zwar Rücken gerade in diesem Bereich Personen gelegentlich in den Hintergrund und der überwiegende Teil des Netzwerkverkehrs wird ausgelöst, ohne das Nutzende davon etwas mitbekommen. Aber in letzter Instanz sind sowohl Verursachende, Überwachende und forschende Menschen. Menschen haben vielfältige Anliegen, müssen unauffälligen von auffälligem Netzwerkverkehr unterscheiden, versuchen die Beweggründe anderer indirekt zu erschließen, zu kategorisieren und in Taxonomien zu visualisieren.

Konzeptionell

Einordnung anhand der Bedeutung des Kontextes und der begrifflichen Beziehungen. Zusätzlich weiterhin in primäre und sekundäre Informationen unterteilt

Primär Kontextinformationen, die gewonnen werden können, ohne bereits vorhandene Daten zu verwenden oder zu kombinieren [2].

Sekundär Kontext, der durch das Verarbeiten von primärem Kontext erschlossen werden kann. Geschehen kann dies durch die Kombination einzelner Datenpunkte einer oder mehrerer Kategorien oder durch Abfragen weiterer Daten mithilfe der primären Informationen [2].

Betrieblich

Wie schon in der Analyse bereits vorgeschlagener Kategorien, erläutert in Kapitel 2, meint betriebliche Kategorisierung: “Einordnung anhand dessen, wie der Kontext akquiriert, modelliert und behandelt wird ”[23].

Zeit Zu welcher Zeit eine Zugriffsanfrage erfolgt.

Ort Von welchem Ort eine Zugriffsanfrage stammt.

Identität Wer Zugriff auf eine Ressource erfragt.

Aktivität Was in einer Situation passiert oder welche Aktion eine Entität ausführt.

Grund Warum etwas getan wird. In Abbildung 4.1 ist kein primärer Grund vorhanden, da ein Grund für etwas ausschließlich aus anderen Informationen erschlossen werden kann und nicht im Netzwerk vorliegt.

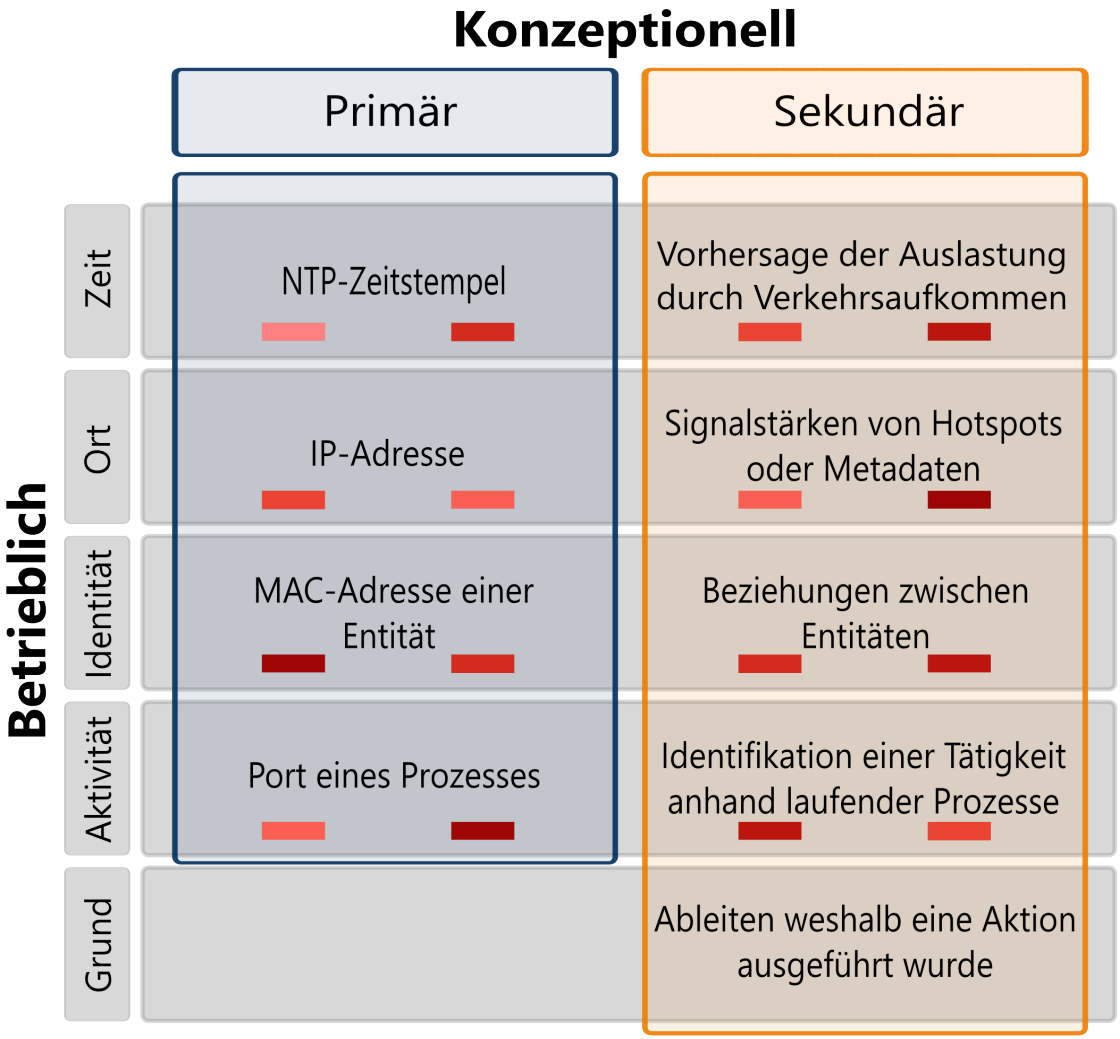

Abbildung 4.1: Nutzerzentrierte Kategorisierung

4.2.2 Netzwerk

Ein Netzwerk besteht im Allgemeinen aus:

1. Entitäten, die daran teilnehmen
2. Kommunikation zwischen den Entitäten
3. Annahmen bzw. Normen bezüglich:
   a) des Verhaltens und der Eigenschaften der Entitäten
   b) der Form und dem Inhalt der Kommunikation

Im Rahmen der Zugriffskontrolle erfolgt Kommunikation im Netzwerk über spezifische Protokolle zwischen verschiedenen Entitäten, die durch bestimmte Attribute charakterisiert werden. Die Normen werden dabei initial festgelegt und im Verlauf der Lebenszeit des Netzwerkes angepasst.

Protokolle

Die Kategorisierung anhand des verwendeten Kommunikationsprotokolls orientiert sich am ISO/OSI-Referenzmodell [7]. Die Zuordnung zu einer bestimmten Ebene und damit Katego-

rie erfolgt anhand der für die einzelnen Schichten üblichen Protokolle. Das ermöglicht die Identifikation von Entitäten anhand der von ihnen genutzten Protokolle. So kann beispielsweise ein Switch oder Router, der nicht von einem Nutzer oder einer Anwendung unterschieden werden.

Anwendung Beinhaltet allen Netzwerkverkehr, der sich der Sitzungsschicht, Darstellungsschicht oder Anwendungsschicht zuordnen lässt.

Transport Pakete, die sich der Transportschicht zuordnen lassen.

Vermittlung Netzwerkverkehr, der zur Vermittlungsschicht gehört.

Entitäten

Kategorisierung von Kontextinformationen abhängig davon, welche Art von Entität sie betreffen. Diese Unterscheidung setzt genauso wie die Historie eindeutig identifizierbare Entitäten voraus.

Gerät Informationen, die sich auf ein spezifisches Gerät beziehen

1. Ein Gerät ist beispielsweise das Endgerät eines Nutzers oder ein Router im Netzwerk.
2. Informationen können beispielsweise die Liste an installierter Software, die Menge laufender Prozesse oder die Auslastung der Hardwarekomponenten sein.

Nutzer/Rolle Informationen, die sich auf einen Nutzer oder seine Rolle beziehen.

1. Ein Nutzer ist dabei eine physische Person und kann mehrere verschiedene Rollen inne haben.
2. Informationen können beispielsweise Zugriffsrechte sein, die mit einer Rolle verbundenen sind. Auch die Browserhistorie eines Nutzers wäre eine Information die in diese Kategorie fällt.

Anwendung Informationen, die sich auf eine Anwendung beziehen.

1. Anwendung umfasst jegliche Softwareprozesse, die für sich oder in Kombination einen bestimmten Zweck erfüllen.
2. Eine Anwendung erzeugt für charakteristischen Netzwerkverkehr, versendet oder empfängt also bestimmte Informationen nach spezifischen Mustern.

Policies

Einordnung der Kontextinformationen abhängig davon, ob sie der für den Anwendungsfall definierten Normen entsprechen.

Erwartet Form der Kommunikation mit verschiedenen Protokollen oder Verhalten von Entitäten entsprechend den im Netzwerk geltenden Vorschriften.

Ungewöhnlich Form der Kommunikation mit verschiedenen Protokollen oder Verhalten von Entitäten, die in Kombination mit den im Netzwerk vorherrschenden Bedingungen entweder auffällig oder irrational sind.

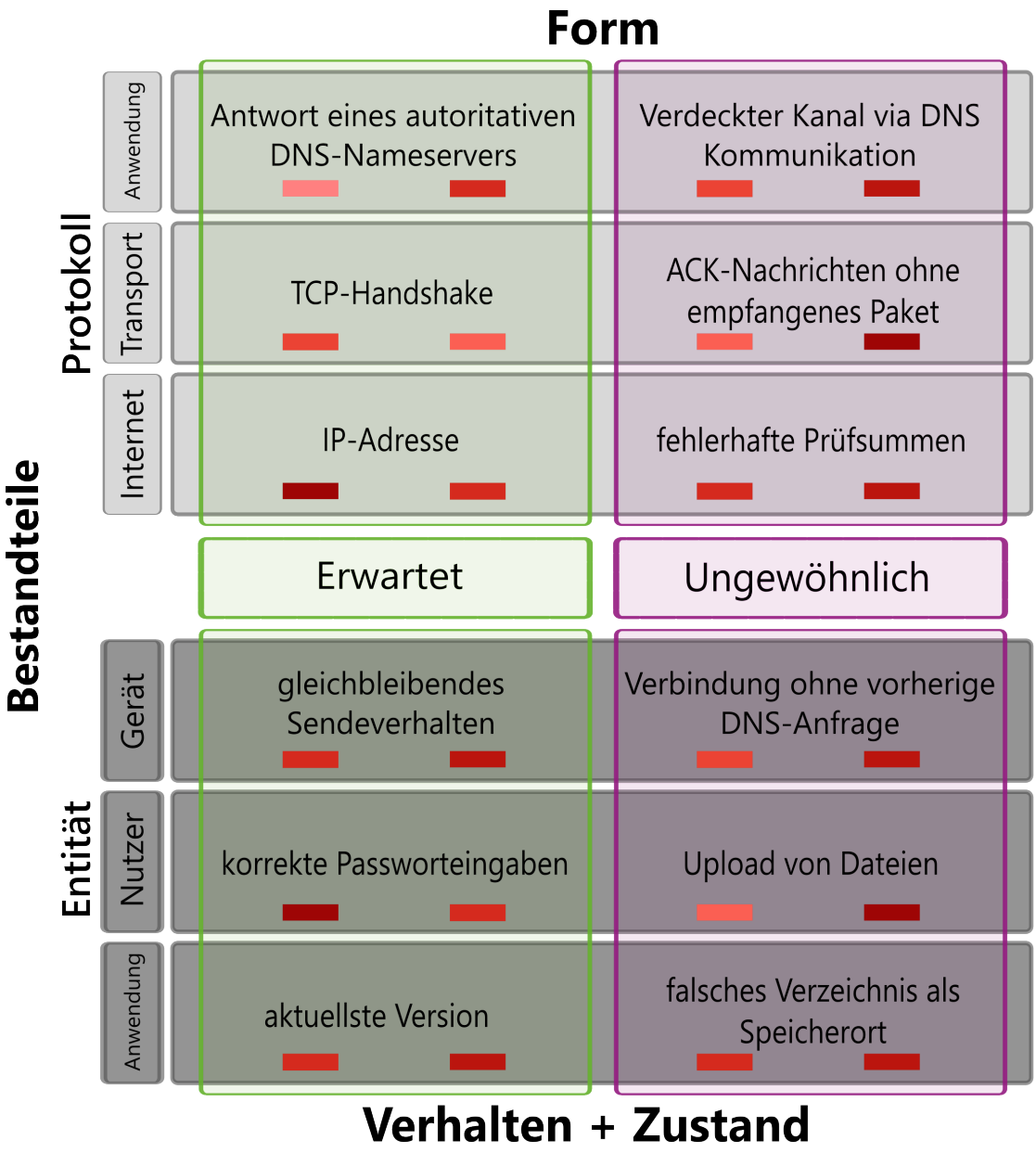

Abbildung 4.2: Kategorisierung von Kontextinformationen als Netzwerkbestandteile

4.2.3 Historie

Unabhängig vom gewählten Fokus oder Blickwinkel auf die Kategorisierung von Kontext benötigt man, um bereits gesammelte Kontextinformationen nutzen zu können, eine Art Gedächtnis. Im Fall eines IDS, welches ohnehin aufgrund seiner Funktionsweise über einen längeren Zeitraum Informationen verarbeitet und neue Erkenntnisse speichert, bietet sich eine Historie, ähnlich wie in Kapitel 2 beschrieben, an. Die Historie einer Entität wird in dieser Taxonomie zweigeteilt. Sie besteht aus:

1. Einem aktiven Teil also ihrem Verhalten, beispielsweise früheren Verbindungen bzw. Verbindungsanfragen
2. Einem passiven Teil also dem Zustand der für sie charakteristischen Attribute, wie etwa einem Nutzernamen oder die Versionsnummer eines bestimmten Programms.

Dynamik

Es liegt in der Natur der Sache, das sich die Messwerte, aus denen sich die Historie einer Entität zusammensetzt, je nachdem welchem Teil sie zugeordnet werden, verschieden oft ändern. Statische Messgrößen ändern dabei ihre Werte nie oder nur sehr selten. Dynamische Messgrößen hingegen sehr oft. In diesem Fall wird eine Unterteilung in jährlich, monatlich, wöchentlich, täglich, stündlich, minütlich und sekündlich vorgenommen.

Raten

Die Historie einer Kontextinformation ist weiterhin in drei verschiedene Bereiche unterteilt. Abhängig davon, wo die Änderung auftritt und ob das IDS oder eine Entität die Aktualisierung des Wertes auslöst.

Änderungsrate Häufigkeit mit der eine Werteänderung im Normalfall vorkommt.

Abtastrate Wie oft Kontextinfomationen abgefragt und aktualisiert werden.

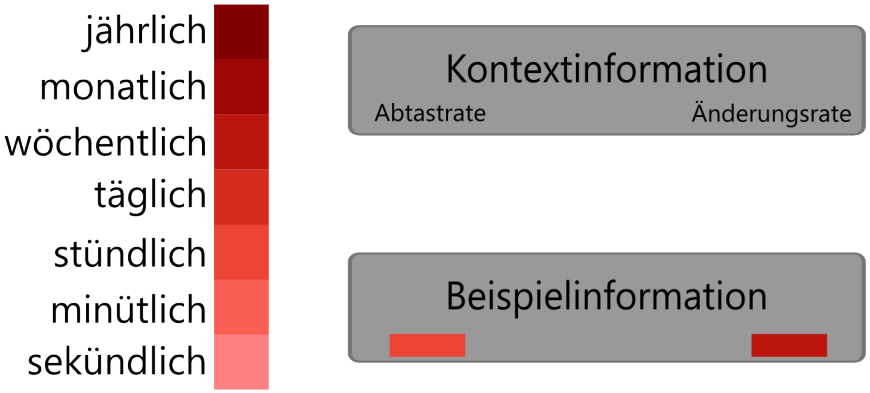

Abbildung 4.3: Legende der Dynamik der Legende

4.3 Form des Kontextes

Nachdem die einzelnen Kategorien definiert wurden, muss festgelegt werden in welcher Form die Kontextinformationen vorliegen sollen bzw. gebracht werden müssen. Für fast alle Kategorien selbsterklärend. Protokollkontext hat eine durch das Protokoll selbst vorgegebene feste Form, ein Zeitstempel ebenso. Kurzgesagt:Größtenteils ist anhand der Spezifikation der jeweiligen eingeordneten Informationen offensichtlich welche Form die Information haben sollte und die Form des Kontextes lediglich nebensächlich. Deshalb wird an dieser Stelle nicht auf alle Kategorien separat eingegangen.
Die Historie bildet eine Ausnahme. Hier gibt es je nach Bedarf des Anwendungsfalls einen

gewissen Spielraum. Die gespeicherten Informationen werden abhängig von der Implementierung eventuell nicht vom IDS eingelesen oder gar selbst vom IDS erzeugt. Auch gibt es höchst unterschiedliche Ansprüche an Form, Umfang und Zeitraum oder Beschränkungen hinsichtlich Performanz und Speicherbedarf.

4.3.1 Historie

Die Historie muss genug Informationen enthalten, um damit neue Urteile in der Gegenwart oder Zukunft zu fällen und sie eindeutig identifizieren zu können. Schließlich kann eine Historie, die dabei helfen soll, Entscheidungen zu treffen, die den Netzwerkverkehr von Entitäten betreffen kann nicht ohne eindeutig identifizierte Entitäten funktionieren.
Die Identifikation erfolgt anhand der Headerinformationen der einzelnen Kommunikationsschichten. In der Historie gespeichert werden entweder die gesamte Payload eines Pakets oder zumindest ein ausreichend aussagekräftige Teilmenge. Die Regelmäßigkeit mit der die Historie aktualisiert bzw. erweitert wird ist von den jeweiligen Gegebenheiten und Anforderungen abhängig. Kommunikationspartner benötigen eine Grundmenge an Informationen um miteinander kommunizieren zu können. Genauso benötigt ein IDS eine bestimmte Mindestmenge an Informationen, um initial eine Entscheidung treffen zu können. Die nach Ansicht des Autors dafür benötigten Informationen finden sich in Tabelle 4.1.

Tabelle 4.1: Benötigte Informationen der Historie einer Entität

Identifikation	Rekonstruktion der ursprünglichen Zugriffsentscheidungen
MAC-Adresse	MAC-Adresse des Ziels
IP-Adresse	IP-Adresse des Ziels, gesetzte Flags, Protokoll, Gültigkeitsdauer
Port	Port des Ziels, protokollspezifische Informationen

Wenn nicht wenigstens diese Information im Netzwerkverkehr enthalten sind, kann ein IDS eine Entität weder eindeutig zuordnen noch eine fundierte Entscheidung treffen. Da also mindestens diese Datenlage bereits benötigt wird, erscheint sie auch als sinnvolle Mindestanforderung für eine Historie. Ohne diese Daten kann keine ausreichende Rekonstruktion des Kenntnisstandes, mit dem die Zugriffsentscheidungen ursprünglich getroffen wurden, ermöglicht werden. Damit ist also keine Aussage über vergangenes Verhalten einer Entität und dessen Bewertung möglich.

4.4 Kontextgewinnung

Jegliche Kategorisierung von Kontext ist von geringem Nutzen, ohne das Informationen vorhanden sind, die kategorisiert werden können. Dabei bedacht werden sollte, an welcher Stelle und wie oft Informationen abgerufen oder automatisch aktualisiert werden müssen [18]. Pimenidis et al. [19] nennen KMDBs, Scanner oder Crawler als Möglichkeiten Kontextinformationen zu sammeln. Zusätzlich dazu sind Anwendungen die direkt auf Endgeräten laufen und von dort aus Daten an das IDS schicken erwähnenswert. Eine Übersicht und genauere Erklärung der einzelnen Komponenten findet sich in Tabelle 4.2.

Tabelle 4.2: Akquirierungspunkte für verschiedene Kontextinformationen

Werkzeug	Erklärung
Konfigurations-management-datenbank (KMDB)	Beinhaltet die Konfigurationsparameter aller bekannten Hosts, einschließlich der vollständigen Historie, mit Details zu Hardware, Software, Prozessen, Infrastrukturen, Verantwortlichkeiten und anderen Komponenten.
Scanner für Netzwerkschwachstellen	Scannen von beispielsweise Ports um sich einen Überblick über laufende Anwendungen und mögliche Schwachstellen zu verschaffen.
Crawler	Erlaubt es die auf einem einzelnen Port aktiven Anwendungen zu ermitteln.
Endpunktagent	Ermöglicht das Abfragen verschiedenster Systemparameter direkt vom Hostsystem. Die Menge an abfragbaren Informationen hängt dabei stark von der gewählten Implementierung ab.

Je nach Wichtigkeit des jeweiligen Hosts variiert der Detailgrad der Informationen die in einer KMDB vermerkt sind. Bei kritischen Infrastrukturen wie Routern, Switches und zentralen Servern sollte die KMDB mindestens das Betriebssystem, dessen Version sowie alle Anwendungen mit ihren Versionen und Patch-Ständen enthalten. Bei allen hier aufgelisteten Werkzeugen sollte bedacht werden, dass aufgrund der Fehleranfälligkeit dieser Herangehensweise Daten zu sammeln, die gewonnenen Informationen in jedem Fall von Menschen gegen geprüft werden sollten.

4.5 Umwandlung der Taxonomie in IDS-Signaturen

Um die im Abschnitt Kontexttaxonomie festgelegten Kategorien in für ein IDS nutzbare Form zu bringen, gilt es gewisse Dinge zu beachten:
die Kontextsensitivität eines Computers unterscheidet sich drastisch von der eines Menschen. Rechensysteme sind sehr gut darin, Daten zu erfassen und zu sammeln, aber Menschen sind immer noch nötig, um verschiedene Kontexte zu erkennen und zu entscheiden, welches Handeln in einer bestimmten Situation angemessen ist [8]. Der limitierende Faktor des Potenzials eines kontextsensitiven Systems ist das Maß an Kontext, das ein Entwickler vorhersehen und codieren kann. Es ist aber weder beim Design noch später bei der Implementierung unmöglich, alle Zusammenhänge vorherzusehen. Dementsprechend schwer wird ist es, ein in sich geschlossenes und allumfassendes Regel-Set festzulegen [18]. Nach Greenberg et al. [10] gibt es drei non-triviale Hauptaspekte, die beim Entwerfen eines kontextsensitiven Systems beachtet werden sollten:

1. Spezifizieren aller möglichen Kontextzustände
2. Wissen welche Informationen einen konkreten Kontextzustand akkurat festlegen.
3. Welche Aktion im jeweiligen Zustand ausgeführt werden sollen.

4.5.1 Entscheidung für ein IDS

IDS können anhand der überwachten Plattform, der verwendeten Erkennungsmethode und der Struktur, in der sie eingesetzt wie in Tabelle 4.3 zu sehen kategorisiert werden [17].

Tabelle 4.3: Einteilungsansätze für IDS

Eigenschaft	IDS Typ	Beschreibung
Plattform	Host	Überwacht Aktivitäten auf dem System, auf dem es eingesetzt wird, um lokale Angriffe zu erkennen.
	Netzwerk	Überwacht Aktivitäten im Netzwerk um Angriffe, die über eine Netzwerkverbindung ausgeführt werden, zu erkennen.
	Hybrid	kombiniert host- und netzwerk-basierte Intrusion Detection Systeme.
Angriffserkennung	Signatur	Überwacht System- und/oder Netzwerkaktivitäten anhand einer Reihe von Signaturen bekannter Angriffe auswertet. Daher ist es nicht in der Lage, Zero-Day-Angriffe zu erkennen, d. h. Angriffe, die Schwachstellen ausnutzen, die vor der Ausführung der Angriffe nicht öffentlich bekannt sind.
	Anomalie	Verwendet ein Basisprofil regulärer Netz- und/oder Systemaktivitäten als Referenz, um zwischen regulären und auffälligen Aktivitäten zu unterscheiden, wobei Letztere als Angriffe behandelt werden. Wird typischerweise durch die Überwachung regulärer Aktivitäten trainiert, um ein Basisaktivitätsprofil zu erstellen.
	Hybrid	Verwendet sowohl signatur-basierte als auch anomalie-basierte Angriffserkennungsmethoden.
Struktur	Zentral	Kann nur an einem einzigen Standort eingesetzt werden.
	Verteilt	Besteht aus mehreren Teilsystemen, für die an verschiedenen Standorten eingesetzt werden können und miteinander kommunizieren, um für die Erkennung von Angriffen relevante Daten, z. B. Angriffswarnungen auszutauschen. Kann koordinierte Angriffe auf mehrere Standorte in einer bestimmten zeitlichen Abfolge erkennen.

Schwächen der verfügbaren IDS

Der Hauptnachteil eines signatur-basierten IDS ist ausschließlich Angriffe zu erkennen, die vordefinierten Verhaltensmustern entsprechen. Der Hauptnachteil eines anomalie-basierten IDS ist die Baseline. Zuerst muss eine Baseline festgelegt werden, was mitunter je nach Größe und Struktur des Netzwerkes sehr komplex und damit fehleranfällig sein kann. Zusätzlich muss diese Baseline mitunter sehr oft aktualisiert werden, um auf Änderungen im Netzwerk zu reagieren. Dies verlangt dem Überwachenden zusätzliche Ressourcen ab.

4.6 Anspruch an das IDS

Um die Performance des konkret gewählten IDS verbessern zu können, muss festgelegt werden, welche Kriterien die Leistung beeinflussen bzw. bestimmen und wie man diese gewichtet. Mell et al. [16] geben eine Übersicht über verschiedene Charakteristiken zur quantitativen Bestimmung der Erkennungsgenauigkeit eines IDS und erläutern zusätzlich die Wechselwirkungen zwischen einzelnen Kriterien, die beim Vergleich verschiedener IDS-Lösungen

beachtet werden sollten.

Abdeckung

Gibt an, welche Typen von Angriffen ein IDS unter idealen Bedingungen erkennen kann.

Wahrscheinlichkeit falscher Alarme

Gibt die Wahrscheinlichkeit das durch ein IDS ausgelöste Alarme durch gutartigen bzw. nicht-schädlichen Netzwerkverkehr verursacht wurden, an.

$$\textit{Rate an falsch – Positiven Meldungen} = \frac{\textit{Anzahl falscher Alarme}}{\textit{Anzahl aller Alarme}}$$

Wahrscheinlichkeit einer Erkennung

Gibt die Rate der durch das IDS korrekt erkannten Angriffe an.

$$\textit{Erkennungswarscheinlichkeit} = \frac{\textit{Anzahl korrekt erkannter Angriffe}}{\textit{Anzahl aller Angriffe}}$$

Resistenz

Ein auf Signaturen basierendes IDS bzw. der menschliche Administrator hinter dem System weisen Probleme auf, die nicht direkt beim Umgang mit verarbeitetem Netzwerkverkehr, sondern schon bei der bewussten, unbewussten oder erzwungenen Entscheidung, welcher Netzwerkverkehr überhaupt infrage kommt, entstehen:

1. Eine zu große Menge an zu verarbeitendem Netzwerkverkehr, die die Verarbeitungskapazität eines IDS übersteigt, kann dazu führen, das Netzwerkpakete verworfen und Angriffe nicht erkannt werden
2. Pakete die zwar nicht bösartig sind, aber so konstruiert das sie möglichst viele IDS Alarme auslösen, überfordern den Administrator oder stören eventuell sogar die Verarbeitung von Paketen generell.
3. Ein Angreifer könnte eine Vielzahl "harmloserer", aber trotzdem noch als schädlich zu deklarierende Pakete senden, um einen größeren Angriff im Netzwerkverkehr zu verschleiern.
4. Pakete, die möglicherweise vorhandene Fehler im IDS selbst ausnutzen.

Korrelation zwischen Einzelereignissen

Demonstriert wie gut ein IDS eine Korrelation zwischen einzelnen Events, möglicherweise verschiedenen Ursprungs herzustellen. Die Ereignisse können dabei aus Routern, Firewalls, Anwendungen, dem IDS selbst oder einer großen Bandbreite anderer Quellen stammen.

Vorhersehen von unbekannten Angriffen

Gibt an, wie gut ein IDS einen Angriff erkennt, der so noch nicht aufgetreten ist. Signaturbasierte IDS sind allgemein mit wenigen Ausnahmen nicht in der Lage, solch einen Angriff zu erkennen. Normalerweise erhöht die Fähigkeit eines Systems, einen noch unbekannten Angriff zu erkennen, im Vergleich zu Systemen, die dies nicht versuchen, zusätzlich die Rate an falsch-positiven Meldungen.

Identifizieren von Angriffen

Wie gut ein IDS einem Angriff, den es erkennt, einen Namen oder eine Kategorie, beispielsweise ein CVE-Nummer, zuordnen kann.

Beurteilung von Angriffen

Indikator dafür, ob ein IDS den Erfolg und die Auswirkungen eines Angriffes korrekt beurteilen kann. In aktuellen Netzwerkumgebungen schlagen viele Angriffs(-vesuche) fehl. Die meisten IDS unterscheiden allerdings nicht zwischen erfolgreichen und fehlgeschlagenen Angriffen. Für denselben Angriff können manche IDS die Anzeichen dafür, ob ein Angriff erfolgreich war erkennen andere lediglich, das ein Angriff stattgefunden hat, allerdings ohne feststellen zu können, ob er erfolgreich war. Die Fähigkeit, den Grad, zu dem ein Angriff auf das überwachte System erfolgreich war, zu beurteilen, ist essenziell. Eine Vorfilterung der Meldungen durch das IDS vereinfacht die Arbeit des Netzwerkadministrators bzw. Analysten stark, da so eine Analyse des Angriffsszenarios und der Korrelation einzelner Angriffe vereinfacht wird. Diese Fähigkeit, bei einem gegebenen IDS messen zu können, setzt das Wissen darüber welche Angriffe erfolgreich sind und welche nicht voraus.

Einordnung der Kriterien

Die Wahrscheinlichkeit, einen Angriff zu erkennen, variiert mit der Rate an falsch-positiven Meldungen. Die Verbesserung der Rate von falsch-positiven Meldungen und Erkennungsrate stehen sich diametral gegenüber. Deutlich wird dies am jeweiligen Extremfall. Erkennt das System den gesamten Netzwerkverkehr als schädlich so erkennt es auch alle Angriffe korrekt. Die Erkennungsrate ist also optimal. Die Falsch-positiv-Rate hingegen sehr schlecht. Erkennt das System kein Paket als Angriff, gibt es keine falsch-positiven Meldungen. Allerdings ist damit auch die Erkennungsrate maximal schlecht. Bei der Konfiguration eines IDS kann also nur in Hinsicht auf eine der beiden Metriken optimiert werden.

4.6.1 Auswahl der Kriterien

Diese Arbeit beschäftigt sich ausschließlich mit sicherheitsrelevanten Leistungsmetriken. Dabei soll hauptsächlich der Einfluss von Kontextsensitivität auf die Abdeckung, die Erkennungswahrscheinlichkeit und die Rate falsch-positiver Alarme betrachtet werden. Ergänzend wird beleuchtet, ob zusätzlicher Kontext die Interpretierbarkeit von Meldungen durch einen menschlichen Nutzer und das Identifizieren von Angriffen verbessert.

5 Implementierung

Der im Kapitel 4 dargelegte Aufbau eines Netzwerkes findet sich auch in der Implementation wieder. Scapy, Wireshark und Zeek verarbeiten Kommunikationsdaten, zeek-agent stellt die Attribute der Entitäten bereit und der Skript-schreibende legt die geltenden Normen durch die erstellten Skripte und das dadurch abgedeckte Verhalten fest.

5.1 Erläuterung der wichtigsten Komponenten

Eine kurze Vorstellung der wichtigsten Komponenten der Implementierung, insofern sie im Rahmen der einzelnen Schritte des Versuchsaufbaus essenziell sind.

5.1.1 Zeek

Ein passives, quelloffenes Analysewerkzeug für Netzwerkverkehr, das via eigener Skriptsprache unter anderem folgendes ermöglicht:

1. Implementierung beliebiger Analyseaufgaben
2. Interpretation von Netzwerkverkehr
3. Erstellung von Protokollen auf der Grundlage dieses Verkehrs

Zeek ist dabei weder rein signatur-basiert wie Suricata noch ausschließlich als Protokollanalysator im Sinne von Wireshark zu verstehen. Vielmehr handelt es sich bei Zeek um eine Mischung, die eine kompakte, aber dennoch detailgetreue Darstellung von Netzwerkprotokollen ermöglicht. Dies erlaubt ein besseres Verständnis des Netzwerkverkehrs und der Netzwerknutzung [1].

5.1.2 Zeek-Agent

Zeek-Agent ist ein Endpunkt-Agent, der Informationen für zentrales Monitoring an Zeek sendet. Abfragen kann man verschiedene Aktivitäten eines Hostsystems, darunter zum Beispiel aktuell laufende Prozesse, offene Sockets oder den Inhalt bestimmter Dateien. Diese erscheinen in Zeek, genauso wie Netzwerkaktivität als Ereignisse und können so in Skripten verwendet werden [26].

5.2 Versuchsaufbau

Der Ablauf für alle Anwendungsfälle ist grundsätzlich sehr ähnlich:

1. Erzeugung
2. Mitschnitt via WireShark
3. Analyse mittels Zeek und Zeek-Agent
   a) Einlesen von Netzwerkverkehr
   b) Einbindung des zusätzlichen Kontextes
   c) Logging

5.2.1 Erzeugung

Der Netzwerkverkehr wurde mit Hilfe von Scapy generiert. Das ermöglicht den für die verschiedenen Szenarien benötigten Netzwerkverkehr zu erzeugen und die einzelnen Schichten eines Pakets an den jeweiligen Anwendungsfall anzupassen. In ist dieser Prozess ausschnittsweise dargestellt. Eine Übersicht über alle dafür verwendeten Skripte findet sich im Anhang.

Quelltext 5.1: Konfiguration und Versendung eines Pakets

```
6  def send_packet(ip_address_src,ip_address_dst):
7      source_server = ip_address_src
8      target_server = ip_address_dst
9      layer_2 = Ether()
10     layer_3 = IP(src=source_server, dst=target_server)
11     layer_4 = TCP(sport=80,dport=43468)
12     tcp_pkt = layer_2 / layer_3 / layer_4
13     sendp(tcp_pkt)
```

5.2.2 Logging

In Zeek erfolgt das Schreiben in Logs immer nach demselben Prinzip:

1. Vor dem Ausführen eines Skriptes wird ein Log und die darin zu speichernden Informationen festgelegt (Z. 3-10)
2. Nutzer-definierter Log wird initialisiert (Z. 19)
3. Form des Eintrags für den Log wird definiert (Z. 20)
4. Log-Eintrag wird der Verbindung als Information hinzugefügt (Z. 23)
5. Eintrag wird in Log geschrieben (Z. 24)

Quelltext 5.2: Generierung einer Log-Datei mit Verbindungsinformationen

```
1  export {
2      # Create an ID for our new stream. By convention, this is called "LOG".
3      redef enum Log::ID += { LOG };

5      # Define the record type that will contain the data to log.
6      type Info: record {
```

```
7         timestamp: time  &log;
8         id: connection_id &log;
9         notice: string  &log;
10     };
11 }

13 redef record connection += {
14    # By convention, the name of this new field is the lowercase name
15    # of the module.
16    examplelog: Info &optional;
17 };
18 event zeek_init(){
19   Log::create_stream(ExampleModule::LOG, $columns=Info, $path="examplemodule"]);
20   local record: ExampleLog::Info = $ts=current_time(), $id=c$id,
         $notice="Example Notice"];
21    # Store a copy of the data in the connection record so other
22    # event handlers can access it.
23    c$examplelog = record;
24    Log::write(ExampleModule::LOG, rec);
25 }
```

5.3 Skripte

Der Kern der Implementierung sind Zeek Skripte. Hier werden die gesammelten Kontextinformationen verwendet. Nachfolgend werden für einige ausgewählte Kategorien Anwendungsfälle vorgestellt.

5.3.1 Geografische Koordinaten und Ortszeit

Das Volumen von durch Menschen verursachten Netzwerkverkehr ist in der Regel größtenteils tageszeitabhängig. So wird üblicherweise nachts weniger kommuniziert als am Tag. Deshalb erscheint es sinnvoll, den Grenzwert für erzeugtes Verkehrsaufkommen, ab dem ein Sender als potenziell böswillig eingestuft wird, je nach Uhrzeit anzupassen.

Das Skript ordnet eine IP-Adresse zu einem geografischen Ort zu (Z. 32).

Berechnet die Ortszeit am Ursprung der Anfrage mithilfe der lokalen Uhrzeit in Kombination mit dem aus dem Abstand ermittelten Zeitunterschied (Z. 36 - 38).

Passt des Grenzwertes entsprechend an (Z. 41).

Quelltext 5.3: Geolokalisierung und Setzen des Grenzwertes

```
function geolocation(c: connection):double{
  local origin_longitude = lookup_location(c$id$orig_h)$longitude;
  return origin_longitude;
}

function time_at_geolocation(longitude: double): int{
  local time_to_add = (longitude - home_longitude)*240;
  return useable_time_at_origin;
}

function set_threshold(c_time: int): double{
  if(c_time< opening_time || c_time > closing_time )
    threshold = threshold-night_time_decrease;
  else
    threshold = threshold+day_time_increase;
  return threshold;
}
```

5.3.2 Verwendete Ports

Wenn eine Verbindung oder ein Verbindungsversuch mit einem bestimmten Port des Systems als Ziel beobachtet wird, ohne das ein Prozess gibt, der diesem Port durch Zeek-Agent zugeordnet werden kann, wird die Verbindung oder der Verbindungsversuch im dazugehörigen Log vermerkt.

Das Skript erfagt bei den zeek-agents in regelmäßigen Abständen die auf Hostsystemen laufenden Prozesse und deren verwendete Ports.

Das Skript vergleicht den Zielport jeder eingehenden Verbindung mit der Liste von verwendeten Ports auf dem Hostsystem.

Abhängig vom Ergebnis wird eine Meldung in den Log geschrieben.

Quelltext 5.4: Abfrage und Abgleich der Ports

```
function check_outgoing_connection(c:connection){
   local _port = port_to_count(c$id$orig_p);
   if(_port !in local_ports){
      local rec: Querytest::Info = [$ts=current_time(), $id=c$id, $notice="No
          Application running on this port"];

event users_result(ctx: ZeekAgent::Context, data: Columns){
   local new_entry : count;
   local connection_port = data$remote_port;
   new_entry = connection_port;
   local_ports[new_entry] = data$name;
}

   local test_query_join = ZeekAgent::query([$sql_stmt=str_stmt_join,
       $event_=query_event, $schedule_=_schedule]);
```

5.3.3 DNS-Auflösung

Menschliche Nutzer verwenden bei der Nutzung ihres Endgerätes im Gegensatz zu Computern keine IP-Adressen, um im Suchanfragen zu formulieren. DNS ordnet IP-Adressen menschenfreundliche Domainnamen zu. Wenn das System eines Nutzers eine Verbindung aufbaut, geht dem eine DNS-Anfrage von diesem Gerät an einen DNS-Nameserver voraus oder die IP-Adresse ist in lokalen Konfigurationsdateien auffindbar. Wenn beispielsweise eine TCP-Verbindung eines Webbrowsers beobachtet wird, ohne das eine zuordenbare DNS-Anfrage erfolgt ist oder die IP-Adresse in der Routingdatei des Endgerätes vermerkt ist, ist das in den meisten Fällen ein Grund zum Handeln.

Das Skript erfragt periodisch die Hosts-Datei eines Unix-Systems (Z. 65).

Loggt parallel für jede Antwort eines DNS-Servers die aufgelöste URL und dazugehörige IP-Adresse (Z. 57).

Im Log vermerkt sind ausgehende Verbindungen, deren Zieladressen vorher nicht durch einen DNS-Server oder die Hosts-Datei aufgelöst wurden (Z. 70).

Quelltext 5.5: Überprüfung der Verbindungsziele eines Endgerätes

```
53 event DNS::log_dns(rec:DNS::Info){
54    local query : string = rec$query;
55    local answer : vector of string = rec$answers;
56    local answer_address = to_addr(answer[1]);
57    resolved_addresses [answer_address] = query;
58 }

60 event query_result(ctx: ZeekAgent::Context, data: Columns){
61 function query_hosts_file(){
62    local str_stmt_hosts = "SELECT columns FROM
          files_columns(\"/etc/hosts\",\"$1:text,$2:text\")";
63    local query_event = query_result;
64    local _schedule = 30 secs;
65    local test_query_join = ZeekAgent::query([$sql_stmt=str_stmt_hosts,
          $event_=query_event, $schedule_=_schedule]);
66 }

68 event check_resolve_table(c : connection){
69    local destination_ip = c$id$resp_h;
70    if(destination_ip !in resolved_addresses && destination_ip !in dns_server){
71       local rec: DNStest::Info = [$ts=current_time(), $id=c$id,
             $notice="Connection without Resolve!"];
72       c$dnstest = rec;
73       Log::write(DNStest::LOG, rec);
74    }
75 }
```

6 Evaluation

In diesem Absatz soll begutachtet werden, inwiefern sich die im Kapitel 5 vorgestellten kontextsensitiven IDS-Skripte auf die im Kapitel 4 priorisierten Leistungsmetriken eines IDS auswirken und inwiefern eine Verbesserung dieser Metriken möglich ist. Das Ziel ist also zu evaluieren, ob sich die Leistung eines IDS verbessern lässt, wenn in Zugriffsentscheidungen Kontextinformationen miteinbezogen werden und welche Informationen sich besonders gut dazu eignen. Dazu wurde zuerst analysiert, ob und wie vorhandener Kontext im Bezug auf NSM kategorisiert werden können und welche Kontextinformationen sich zur Anpassung der Arbeitsabläufe eines IDS eignen. Diese sollten zur besseren Übersicht in einer Taxonomie in Beziehung zueinander gesetzt. Zusätzlich wurde erörtert wie und wo diese Informationen zu beschaffen sind. Bewertet werden sollte dabei, inwiefern es Möglichkeiten der Anpassung an sich verändernde Bedingungen gibt und Kontextinformationen zur Verbesserung der Leistung eines IDS beitragen. Das Hauptaugenmerk liegt dabei darauf, falsch-positive Meldungen zu verringern und die Interpretierbarkeit der Logs zu erhöhen, indem zusätzliche Kontextinformationen in die Entscheidungsfindung des IDS miteinbezogen werden. Dazu wurde untersucht, wie gut sich die für den jeweiligen Anwendungsfall benötigten Informationen sammeln und verarbeiten lassen. Anhand dessen soll eine Einschätzung darüber gegeben werden, welche Kontextkategorien sich zur Verbesserung der IDS-Metriken eignen. Dabei wurden Kontextinformationen aus verschiedenen Einzelkategorien der im Kapitel 4 vorgestellten Taxonomie kombiniert, um im jeweiligen Anwendungsfall eine Verbesserung der IDS-Eigenschaften zu erreichen. Die Nützlichkeit der Kontextinformationen hängt dabei von der Verfügbarkeit und Qualität der Informationen sowie der Komplexität, dass dazu benötigte Skript zu erzeugen ab. Die Verfügbarkeit wird daran bemessen wie anspruchsvoll die Informationen zu beschaffen waren. Die Qualität wird von der Höhe des Aufwandes die Informationen in eine Form zu bringen, die eine Verarbeitung zulässt und .

6.1 Bedeutung von Kontextsensitivität

Um die im Kapitel 4 besprochenen Nachteile auszugleichen, bietet es sich an, die bereits verfügbaren, gesammelten und aufbereiteten Kontextinformationen zu verwenden. Dies ermöglicht die schon existenten Signaturen so anzupassen, dass sie eine größere Menge von Ereignissen abdecken oder schon definierte Ereignisse genauer zu spezifizieren und so weniger (falsche) Meldungen zu generieren, ohne dabei das Verhalten des Netzwerkverkehrs dauerhaft neu bewerten zu müssen.

6.2 IDS-Implementierung

Zeek bot im Vergleich zu anderen IDS die beste Software bzw. Umgebung für die Entwicklung neuer kontextsensitiver Erkennungs- oder Verarbeitungstechniken. Es kann für die kontinuierliche Überwachung von Netzwerken mit hohem Durchsatz verwendet werden. Die Skripting-Umgebung ist in einer speichersicheren Sprache erweiterbar, die auf die Verarbeitung von Netzwerkdaten spezialisiert ist. Im Gegensatz zu anderen Tools ist es nicht auf ein einziges Paradigma für die Netzüberwachung limitiert. Aber auch Zeek in Kombination mit Zeek-Agent ist nicht perfekt. Die Vor- und Nachteile der in dieser Arbeit verwendeten Toolchain werden in den nachfolgenden Abschnitten an gegebener Stelle diskutiert.

6.3 Vergleich der Kontextkategorien

Ähnlich wie in den im Bereich 4 verwendeten Grafiken erfolgt auch die Bewertung der Unterteilung im Hinblick auf Verfügbarkeit und Qualität. Auf die im Teilbereich 5 implementierten Rubriken und dabei aufgetretene Problem wird jeweils noch einmal separat eingegangen. Alle weiteren im Kapitel 4 aufgestellten Kategorien werden anhand der Erkenntnisse, die während des Implementationsprozesses entstanden sind, ausschließlich in der Grafik 6.1 bewertet. Dabei erfolgt auch eine Veranschaulichung davon welche Kategorie zur Erfüllung welches Ziels geeignet ist. Dargestellt wird weiterhin, ob eine Kategorie zur Verbesserung der Metriken für sich allein, oder nur in Kombination mit anderen Kategorien beitragen kann.

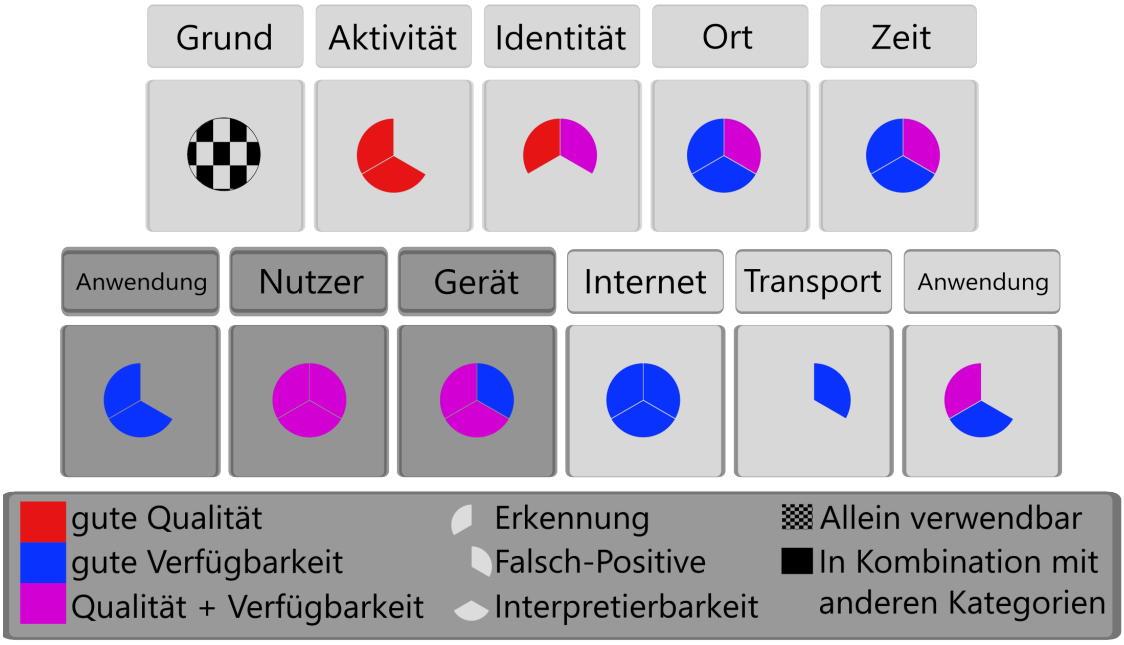

Abbildung 6.1: Evaluation der Taxonomie

6.3.1 Verfügbarkeit der Informationen

Die Informationen, die im Kapitel 4 beispielhaft als gegeben vorausgesetzt werden, sind in der Praxis unterschiedlich schwer zu akquirieren. Es ist beispielsweise um ein Vielfaches einfacher, korrekt die aktuelle Uhrzeit zu ermitteln als den Grund dafür, das ein Nutzer eine Aktion ausführt, zu bestimmen. Auch sind einige Informationen für die Funktionalität des

Netzwerkes und der Kommunikation zwingend notwendig, andere hingegen sind optional oder auf bestimmten Systemen nicht vorhanden.

Anwendungsfälle

Die Informationen, die mittels Zeek-Agent direkt vom Host abgefragt werden, sind, da sie ohnehin zum korrekten Betrieb des Systems in der Praxis gebraucht werden, stets verfügbar. Einzig die Zeit, die eine Zeek-Agent-Instanz für eine Antwort benötigt, hat sich bei komplexeren Anfragen oder zu langsamer Hardware als problematisch erwiesen. Es ist also weniger ein Problem, dass Informationen generell nicht vorhanden sind, sondern lediglich nicht ausreichend schnell zur Verfügung gestellt werden können. Im Fall der verwendeten Ports (5.4) führte die Komplexität der an den Zeek-Agent gesendeten Anfrage dazu, das eine Antwort mitunter erst eintraf, nachdem schon zum Hostsystem zugehöriger Netzwerkverkehr bearbeitet wurde. Dies kann vor allem in Netzwerken mit einem hohen Durchsatz und langsam antwortenden Hostsystemen zu fälschlicherweise geblocktem Verkehr führen. Zu einem gewissen Grad lies sich diese Problematik mit festem Scheduling einzelner Events umgehen. Das Scheduling verschiedener Ereignisse mit teilweise asynchronen Antworten wirft allerdings neue Probleme auf. Es empfiehlt sich deshalb, statt eines großen komplexen Skripts mehrere kleinere Skripte zu verwenden, um Leistungseinbrüche zu vermeiden. Dieser modulare Ansatz hat außerdem den Vorteil, dass sich die Menge an Ergebnissen, die miteinander in Bezug gesetzt werden können, vergrößert. So können über einen längeren Zeitraum oder mehrere Geräte verteilte Angriffe noch besser erkannt werden, als wenn lediglich Kontextsensitivität allein verwendet wird.

6.3.2 Qualität der Informationen

Auch die Qualität der Informationen ist in der Praxis limitiert. Genauigkeit und Korrektheit der Informationen sind teilweise stark unterschiedlich. Auch die Vertrauenswürdigkeit und Aktualität der Daten variiert. Es kann im Allgemeinen davon ausgegangen werden, dass Informationen, die von einem Host, der sich im eigenen Netz befindet, stammen, nicht manipuliert sind. Netzwerkverkehr, der von außerhalb das Netzwerk erreicht, ist mit einer höheren Wahrscheinlichkeit verfälscht, gerade wenn die verwendeten Kommunikationsprotokolle und Strukturen keine Möglichkeiten bieten, Informationen oder die Seriosität des Senders zu prüfen. Der momentan limitierende Faktor im Fall der Beispielskripte sind die in Zeek-Agent verfügbaren Tabellen. In einer früheren Version des Zeek-Agent bestand die Option, externe Tools wie osquery anzubinden, um eine wesentlich größere Menge an Informationen als zum gegenwärtigen Zeitpunkt zu verwenden. Dieser Ansatz wurde allerdings von den Entwicklern aufgrund der zu hohen Komplexität verworfen. Nichtsdestotrotz deckt Zeek-Agent in Kombination mit Zeek die im Kapitel 4 vorgeschlagenen Kategorien in ihren Grundzügen sofern möglich, weitestgehend ab. Bestenfalls stammen Informationen von einer allgemein vertrauenswürdigen Quelle. Außerhalb des Netzwerkes kann das, wie im Skript 5.5, zum Beispiel ein autoritativen DNS-Nameserver sein. Innerhalb des Systems sind das Informationen, auf die ein Endnutzer ohne erweiterte Rechte nur begrenzten oder keinen Einfluss ausüben kann oder das meist ohne bewusstes Zutun, wie beispielsweise im Skript 5.4 die Wahl des Ports geschieht. Damit ist solch ein Datenpunkt ein guter Indikator, um potenziell auffälligen Netzwerkverkehr zu identifizieren.

6.4 Leistungsverbesserung

Die drei Anwendungsbeispiele haben gezeigt, das ein Kontext sowohl die Erkennung von Angriffen und die Rate an Falsch-positiven als auch die Interpretierbarkeit des Geschehens im Netzwerk verbessert.

7 Zusammenfassung und Ausblick

Es gibt eine große Menge an möglichen Ansätzen Kontext einzuteilen. Einige davon wurden in dieser Arbeit vorgestellt, zwei selbst in Kapitel 4 konstruiert, in Kapitel 5 exemplarisch getestet und anschließend in Kapitel 6 ausgewertet. Es hat sich dabei gezeigt, dass Kategorien, bis auf wenige Ausnahmen, erst in Kombination mit anderen Kategorien ihr volles Potenzial entfalten können. Auch hat selten eine Kategorie allein alle Leistungsmetriken gleichzeitig abdecken können. Zeek hat sich dabei, aufgrund seiner vielfältigen Möglichkeiten, als ein sehr gutes Werkzeug zum Testen der Kategorien herausgestellt. Lediglich die hohe Komplexität von Zeek war an einigen Stellen ein Problem. So war das, für die Verwirklichung einer minimalen Testumgebung, benötigte Setup teilweise umständlicher als gedacht. So hat dieser Teil, in Relation zur Relevanz für die eigentlichen Aufgabe, unverhältnismäßig viel Zeit gekostet und so immer wieder zu Verzögerungen geführt. Nichtsdestotrotz sind die Möglichkeiten die Zeek bietet, wenn ersteinmal verstanden wurde wie sie zu nutzen sind, quasi unbegrenzt. Zeek-Agent hingegen hat sich als weniger mächtig in seinen Möglichkeiten als zunächst vermutet herausgestellt. Positiv zu erwähnen, ist die hervorragende Einbindung in Zeek. Negativ allerdings die, im Vergleich zu anderen solchen Werkzeugen, geringe Anzahl an abfragbaren Informationen. Hier bietet es sich also an eventuell andere, in Kapitel 4, vorgestellte Mittel zu verwenden. Zusammengefasst lässt sich aber sagen, dass mehr Kontextinformationen in den allermeisten Fällen die Leistung verbessern.
Ein in dieser Arbeit nicht betrachteter Aspekt sind Prozesse die in ihrem Lebenszyklus keine Daten auf die Festplatte schreiben oder anderweitig nachweisbar Spuren hinterlassen. Es wurde sich ausschließlich mit Vorgängen beschäfigt, die nachweislich einlesbare Daten erzeugen oder für die Gewährleistung ihrer Funktionalitätzumindest benötigen. Netzwerkverkehr von Schadsoftware, die beispielsweise ausschließlich im Arbeitsspeicher eines Systems arbeitet, kann nicht mit zusätzlichen Kontextinformationen kombiniert werden. Hier wäre eine zukünftige Betrachtung der Erkennungsmöglichkeiten also dringend notwendig, da gerade diese Art von Schadsoftware ein zunehmendes Problem, nicht nur im Bereich der IT-Sicherheit, darstellt und es, nach Wissensstand des Autors, bisher noch keine ausgereiften Ansätze zur bekämpfung gibt.

Literatur

[1] *About Zeek — Book of Zeek (git/master)*. URL: https://docs.zeek.org/en/master/about.html (besucht am 21. 09. 2022).

[2] Gregory D. Abowd u. a. "Towards a Better Understanding of Context and Context-Awareness". In: *Handheld and Ubiquitous Computing*. Hrsg. von Hans-W. Gellersen. Bearb. von Gerhard Goos, Juris Hartmanis und Jan van Leeuwen. Bd. 1707. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, S. 304–307. ISBN: 978-3-540-66550-2 978-3-540-48157-7. DOI: 10.1007/3-540-48157-5_29.

[3] Jose Aguilar, Marxjhony Jerez und Taniana Rodríguez. "CAMeOnto: Context awareness meta ontology modeling". en. In: *Applied Computing and Informatics* 14.2 (Juli 2018), S. 202–213. ISSN: 22108327. DOI: 10.1016/j.aci.2017.08.001.

[4] Unai Alegre, Juan Carlos Augusto und Tony Clark. "Engineering context-aware systems and applications: A survey". en. In: *Journal of Systems and Software* 117 (Juli 2016), S. 55–83. ISSN: 01641212. DOI: 10.1016/j.jss.2016.02.010.

[5] Mary Bazire und Patrick Brézillon. *Understanding Context Before Using It*. en. Hrsg. von David Hutchison u. a. Bd. 3554. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 29–40. ISBN: 978-3-540-26924-3 978-3-540-31890-3. DOI: 10.1007/11508373_3.

[6] Suan Khai Chong u. a. "Context-aware sensors and data muffling". In: *Context awareness for self-managing systems (devices, applications and networks) proceeding* (2007), S. 103–117.

[7] John D Day und Hubert Zimmermann. "The OSI reference model". In: *Proceedings of the IEEE* 71.12 (1983), S. 1334–1340.

[8] Anind K Dey. "Understanding and using context". In: *Personal and ubiquitous computing* 5.1 (2001), S. 4–7.

[9] Ibrahim Ghafir, Jakub Svoboda und Vaclav Prenosil. "Network Monitoring Approaches An Overview". en. In: *Third International Conference on Advances in Computing, Communication and Information Technology- CCIT 2015*. Institute of Research Engineers und Doctors, Mai 2015, S. 118–123. ISBN: 978-1-63248-061-3. DOI: 10.15224/978-1-63248-061-3-72.

[10] Saul Greenberg. "Context as a dynamic construct". In: *Human–Computer Interaction* 16.2-4 (2001), S. 257–268.

[11] Karen Henricksen. “A framework for context-aware pervasive computing applications”. School of Information Technology und Electrical Engineering, The University of Queensland, 2003. URL: http://henricksen.id.au/publications/phd-thesis.pdf.

[12] Anita K Jones und Robert S Sielken. “Computer system intrusion detection: A survey”. In: *Computer Science Technical Report* (2000), S. 1–25.

[13] A. S. M. Kayes, Jun Han und Alan Colman. “ICAF: A Context-Aware Framework for Access Control”. In: ACISP'12. Wollongong, NSW, Australia: Springer-Verlag, 2012. ISBN: 9783642314476. DOI: 10.1007/978-3-642-31448-3_34.

[14] A. S. M. Kayes u. a. “A Survey of Context-Aware Access Control Mechanisms for Cloud and Fog Networks: Taxonomy and Open Research Issues”. en. In: *Sensors* 20.9 (2020), S. 2464. ISSN: 1424-8220. DOI: 10.3390/s20092464.

[15] A.S.M. Kayes, Jun Han und Alan Colman. “An ontological framework for situation-aware access control of software services”. en. In: *Information Systems* 53 (2015), S. 253–277. ISSN: 03064379. DOI: 10.1016/j.is.2015.03.011.

[16] Peter Mell u. a. *An Overview of Issues in Testing Intrusion Detection Systems*. en. 2003. DOI: https://doi.org/10.6028/NIST.IR.7007.

[17] Aleksandar Milenkoski u. a. “Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices”. en. In: *ACM Computing Surveys* 48.1 (2015), S. 1–41. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2808691.

[18] Charith Perera u. a. “Context Aware Computing for The Internet of Things: A Survey”. In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), S. 414–454. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.042313.00197.

[19] Lexi Pimenidis, Benedikt Westermann und Volker Wetzelaer. “A context aware network-IDS”. In: *Proceedings of The 13th Nordic Workshop on Secure IT Systems NordSec 2008*. 2008, S. 3.

[20] Karen Scarfone, Peter Mell u. a. “Guide to intrusion detection and prevention systems (idps)”. In: *NIST special publication* 800.2007 (2007), S. 94.

[21] Bill Schilit, Norman Adams und Roy Want. “Context-aware computing applications”. In: *1994 first workshop on mobile computing systems and applications*. IEEE. 1994, S. 85–90.

[22] Bruce Schneier. “Managed Security Monitoring:Network Security for the 21st Century”. In: *Computers & Security* 20.6 (Sep. 2001), S. 491–503. ISSN: 01674048. DOI: 10.1016/S0167-4048(01)00607-1.

[23] Arthur H Van Bunningen, Ling Feng und Peter MG Apers. “Context for ubiquitous data management”. In: *International Workshop on Ubiquitous Data Management*. IEEE. 2005, S. 17–24.

[24] Wei Liu, Xue Li und Daoli Huang. “A survey on context awareness”. In: *2011 International Conference on Computer Science and Service System (CSSS)*. 2011 International Conference on Computer Science and Service System (CSSS). Nanjing, China: IEEE, Juni 2011, S. 144–147. ISBN: 978-1-4244-9762-1. DOI: 10.1109/CSSS.2011.5972040.

[25] J. Wolfgang Kaltz, Jürgen Ziegler und Steffen Lohmann. “Context-aware Web Engineering: Modeling and Applications”. In: *Revue d'intelligence artificielle* 19.3 (1. Juni 2005), S. 439–458. ISSN: 0992499X. DOI: 10.3166/ria.19.439-458.

[26] *zeek-agent-v2: Open source endpoint agent providing host information to Zeek. [v2]*. URL: https://github.com/zeek/zeek-agent-v2 (besucht am 21. 09. 2022).

[27] Andreas Zimmermann, Andreas Lorenz und Reinhard Oppermann. “An Operational Definition of Context”. In: *Modeling and Using Context*. Hrsg. von Boicho Kokinov u. a. Bd. 4635. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 558–571. ISBN: 978-3-540-74254-8. DOI: 10.1007/978-3-540-74255-5_42.