TECHNISCHE UNIVERSITÄT DRESDEN

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

Bachelor-Arbeit

Thesis title

Student name

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

Betreuer
Supervisor

Betreuender Hochschullehrer
Prof. Dr. Professor

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 das vorliegende Dokument mit dem Titel *Thesis title* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in diesem Dokument angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Während der Anfertigung dieses Dokumentes wurde ich nur von folgenden Personen unterstützt:

Supervisor name

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

Student name

Zusammenfassung

The abstract serves as a summary of the work. It should concisely (around half a page) answer the following questions:

- What is the problem tackled in this thesis?
- Why is this problem relevant?
- What contributions does this thesis contain with regards to the problem?
- Which scientific method(s) was/were used and what were their results?

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 aber zu erläutern oder gar korrekt und vollständig zu definieren, fällt allerdings um ein Vielfaches schwerer [8]. Auch herrscht in verschiedenen Fachbereichen jeder davon mit anderen Sachverhalten und Problemen, die die jeweiligen Autoren zu lösen versuchen, 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, wie solch eine Definition aussehen soll [4, 21] nicht zufriedenstellend möglich. Die Menge an Anwendungsfällen und damit auch die zu beachtenden Gesichtspunkte, die man für eine den Ansprüchen genügenden 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 stammt von Schilit et al.[18] 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 [18] Kontext Attribute wie Beleuchtung, Lautstärke, den Grad der Netzwerkverbindung, Kommunikationskosten, Kommunikationsbandbreite und die soziale Situation. Nach der Definition von Dey [8] 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 meisten anderen Autoren als Quasikonsens akzeptiert [3, 4, 21] oder als Ausgangspunkt für ihre eigene Definition genutzt. [13, 24].
Kaltz et al. [22] verstehen Kontext als ein Kontextraum, also eine Kombination aus Kontextparametern, Elementen einer domänenspezifischen Ontologie und Dienstleistungsbeschreibungen in Form von $C = \{U; P; L; T; D; I; S\}$ definiert. 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.
Bazire und Brézillon [5] 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.
Kayes et al.[13] definieren Kontextinformationen in Bezug auf Zugriffskontrollentscheidungen als relevante Informationen über den Zustand einer Entität (Nutzer, Ressource, Ressourcenbesitzer) und deren Umgebung oder die Beziehung zwischen Entitäten.

2.1.2 Zugriffskontrolle

2.2 Kontextkategorisierung

Nachdem ein Überblick über das Kontextverständnis in der gängigen Literatur gegeben und Kontext im Bezug auf Zugriffskontrolle erläutert wurde, folgt eine Vorstellung ausgewählter Ansätze der Kategorisierung.

2.2.1 Analyse bereits vorgeschlagener Kategorien

Auch bei der Kategorisierung von Kontext gibt es richtungsweisende Vorschläge. Abowd et al. [2] haben einen der führenden Mechanismen zur Definition von Typen von Kontext vorgeschlagen. Sie identifizierten Ort, Zeit, Identität und Aktivität als primäre Kontexttypen. Weiterhin wird sekundärer Kontext als Kontext definiert, der durch Nutzung von Primärkontext erschlossen werden kann. Schilit et al. [18] 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 et al. [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. [20] 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] schlägt Historie als Kontextkategorie vor. Dabei wird die zeitliche Entwicklung einer bestimmten Messgröße in der Vergangenheit als Kontext definiert. Das erlaubt die Festlegung von Standardwerten. Damit ist unter Umständen eine Vorhersage darüber, welche Werte die Messgrößen zukünftig annehmen werden möglich.

3 Voraussetzungen und verwandte Arbeiten

Situation Eine Situation

Umgebung Umgebung ist ein Synonym für Kontext [2].

3.1 Netzwerk-Sicherheits-Monitoring

Ghafir et al. [9] beschreiben Sicherheits-Monitoring und dessen Aufgaben wie folgt: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

Angriffserkennung ist "der 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"[17].

3.1.2 Intrusion Detection System

Nach Jones et al. [12] werden Informationssysteme immer umfassender werden und haben einen stetig höher werdenden Wert für Netzwerksicherheit. Auf Grundlage der Definition von Intrusion Detection ist ein IDS somit Software, die den Prozess, einen Eingriff zu erkennen, automatisiert [17]. Schneier [19] 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

Zuerst wird die Notwendigkeit für eine Taxonomie erläutert. Dann erfolgt die Definition der Taxonomiekategorien. Danach wird festgelegt, in welcher Form die Kontextinformationen in den einzelnen Kategorien vorliegen müssen und wie man 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 vorgeschlagenen Designs und Kontextsensitivität gelöst werden.

4.1 Notwendigkeit einer Taxonomie

Die Evaluation verschiedener Kategorisierungsschemata zeigt, dass keine Kategorisierung allen Ansprüchen gerecht werden kann [16].

4.2 Erstellung einer Taxonomie

Es folgt eine Kombination bereits vorgeschlagener Kategorisierungsschemata. Dazu lege ich zuerst die Definition der einzelnen Kategorien im Bezug auf kontextsensitive Zugriffskontrolle fest. Die Kategorien müssen dafür abhängig davon, mit welchem Fokus sie der jeweilige Autor konstruiert hat, mehr oder weniger stark angepasst werden. Die Grafiken 4.1 und 4.2 veranschaulichen die Beziehung der einzelnen Kategorien zueinander und geben Beispiele dafür, wie Kontextinformationen aussehen können. Als Vorbild diente [16].

4.2.1 Mensch

Die offensichtliche Kategorisierung von Kontext und auch von den meisten Autoren, auf die im Kapitel 2 erwähnt wurden, 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 verpacken.

Konzeptionell

Einordnung anhand der Bedeutung des Kontextes und der begrifflichen Beziehungen.

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

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

Betrieblich

Wie schon in der Analyse bereits vorgeschlagener Kategorien bedeutet die betriebliche Kategorisierung: “Einordnung anhand dessen, wie der Kontext akquiriert, modelliert und behandelt wird ”[20].

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.

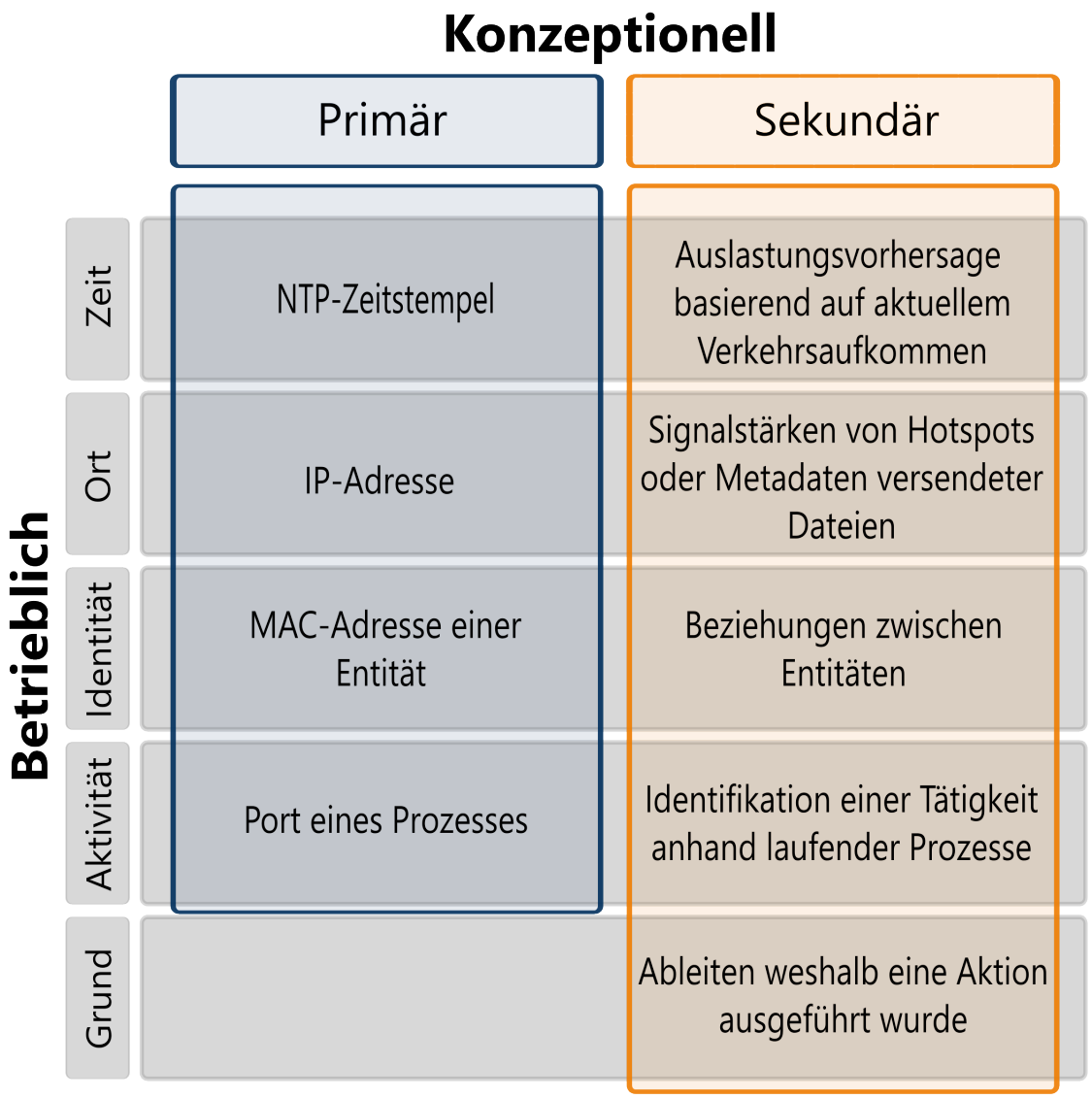

Abbildung 4.1: Nutzerzentrierte Kategorisierung

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

Kategorisierung anhand des verwendeten Kommunikationsprotokolls. Orientiert sich am ISO/OSI-Referenzmodell [7]. Die Zuordnung zu einer bestimmten Ebene und damit Kategorie 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 einen Router oder das Endgerät eines Nutzers.
2. Informationen können beispielsweise die Liste an installierter Software, die Menge laufender Prozesse oder die Auslastung der Hardwarekomponenten sein.

Nutzer Informationen, die sich auf einen Nutzer beziehen.

1. Ein Nutzer im Sinne eines Computersystems und keine physische Person. So kann eine physische Person mehrere verschiedene logische Nutzerkonten haben.
2. Informationen können zum Beispiel die Zugriffsrechte eines Nutzers sein.

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 sie charakteristischen Netzwerkverkehr, versendet oder empfängt also bestimmte Informationen.

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.

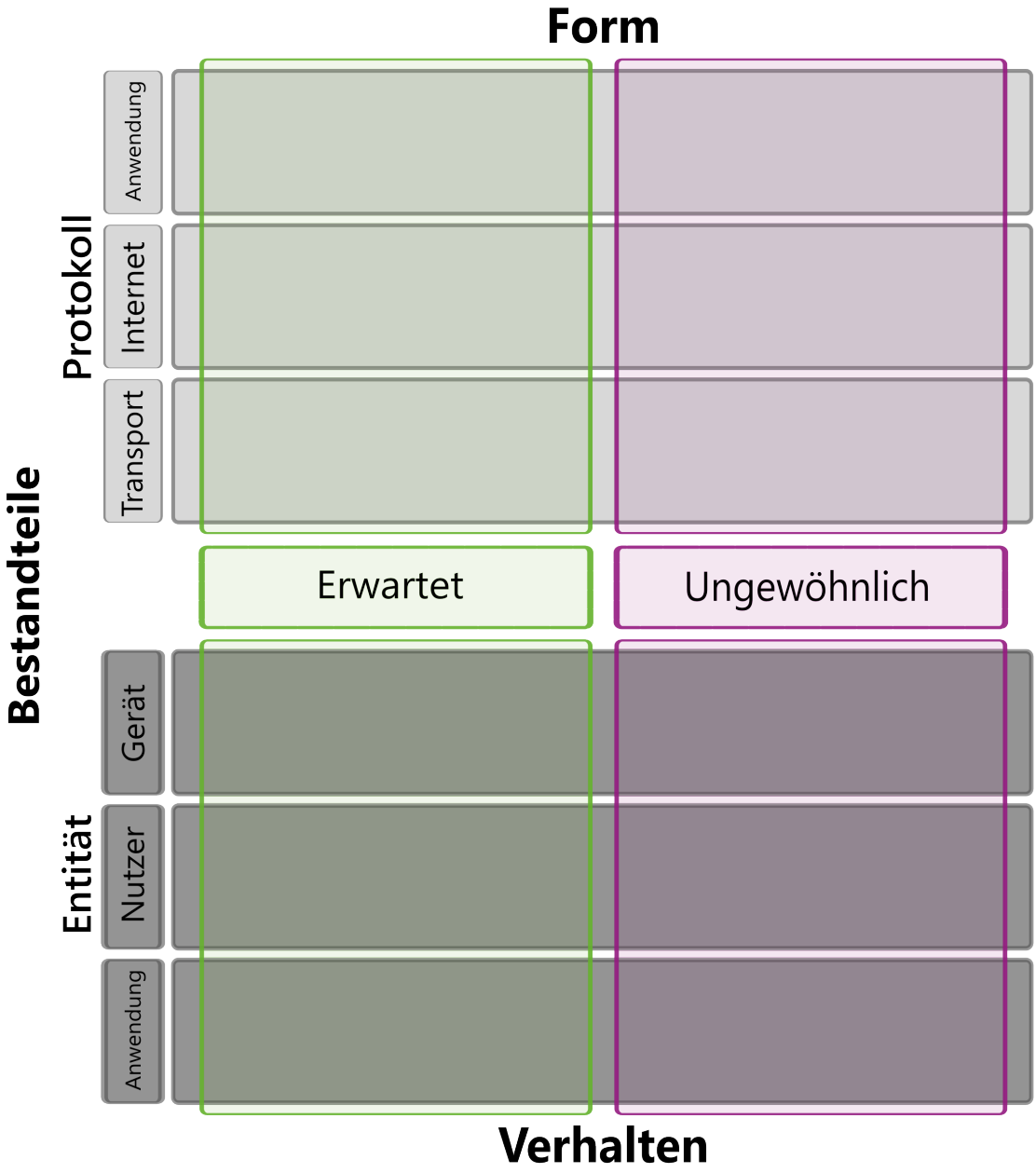

Abbildung 4.2: Kategorisierung von Kontextinformationen als Netzwerkbestandteile

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 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 abändern 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 ist weiterhin in 3 verschiedene Bereiche unterteilt. Abhängig davon, wo die Änderung auftritt und ob das IDS oder eine Entität die Aktualisierung des Wertes auslöst.

Änderungsrate Gibt an, wie oft eine Entität eine Werteänderung mitteilt.

Abtastrate Wie oft Werte einer Entität vom IDS abgefragt bzw. aktualisiert werden.

Abfragerate Wie oft Werte im Netzwerk von einer Entität, die nicht das IDS ist, abgefragt werden.

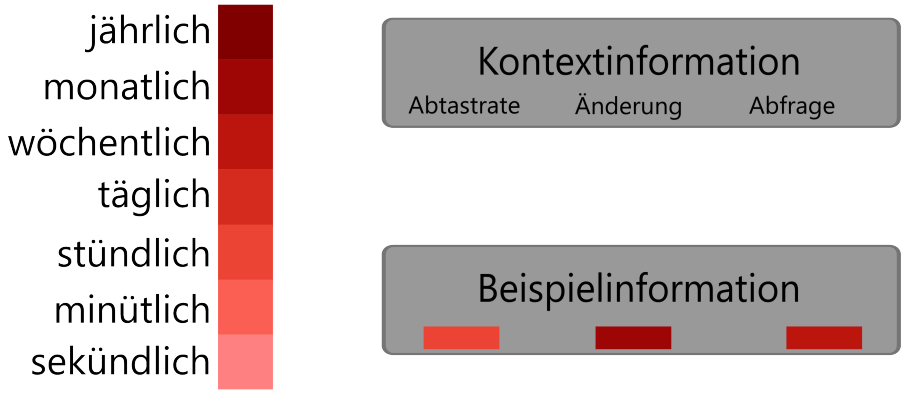

Abbildung 4.3: Farblegende der Dynamik und Historie eines Beispieleintrages

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 seperat 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. Nach Ansicht des Autors sollte ein Teilmenge mindestens folgende Daten beinhalten, um eine ausreichende Rekonstruktion des Kenntnisstandes, mit dem die Zugriffsentscheidungen ursprünglich getroffen wurden, zu ermöglichen und damit Aussagekraft zu gewährleisten:

1. Zur Identifikation einer Entität und der Zuordnung ihrer Verbindung:
   a) MAC-Adresse
   b) IP-Adresse
   c) Port
2. Zur Rekonstruktion der ursprünglichen Zugriffsentscheidungen, die die Entität betreffen:
   a) MAC-Adresse des Ziels
   b) IP-Adresse des Ziels, gesetzte Flags, Gültigkeitsdauer, Protokoll
   c) Port des Ziels, protokollspezifische Informationen

4.4 Kontextgewinnung

Jegliche Kategorisierung von Kontext ist von geringem Nutzen, ohne das Informationen vorhanden sind, die kategorisiert werden können. Die benötigten Informationen entweder direkt aus dem Netzwerkverkehr oder erfragt sie von Dabei sollte man bedenken, an welcher Stelle und wie oft man Informationen abruft oder automatisch aktualisiert. [16]

TODO Es folgt eine Übersicht einiger Beispiele für Möglichkeiten Kontextinformationen zu gewinnen. - osquery - network (open ports) | Nessus - devices in network | Configuration Management Database (CMDB), nmap - cve reference - yes/no - protocol header

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 [16]. 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 4.1 zu sehen kategorisiert werden [15].

Tabelle 4.1: 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.[14] 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.

Unbekannte Angriffe vorhersehen

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 eines Angriffs

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 Voralterung 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 [23].

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 };
4
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 }
12
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.
Zeile 32 Zuordnung einer IP-Adresse zu einem geografischen Ort.
Zeile 36 - 42 Ermittlung der Ortszeit am Ursprung der Anfrage mithilfe der lokalen Uhrzeit in Kombination mit dem aus dem Abstand ermittelten Zeitunterschied.
Zeile 45 Anpassung des Grenzwertes

Quelltext 5.3: Geolokalisierung und Setzen des Grenzwertes

```
31 function geolocation(c: connection):double{
32   local origin_longitude = lookup_location(c$id$orig_h)$longitude;
33   return origin_longitude;
34 }
35
36 function time_at_geolocation(longitude: double): int{
37   local time_to_add = (longitude - home_longitude)*240;
42   return useable_time_at_origin;
43 }
44
45 function set_threshold(c_time: int): double{
46   if(c_time< opening_time || c_time > closing_time )
47     threshold = threshold-night_time_decrease;
```

```
48    else
49      threshold = threshold+day_time_increase;
50    return threshold;
51  }
```

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.

1. Das Skript erfagt bei den zeek-agents in regelmäßigen Abständen die auf Hostsystemen laufenden Prozesse und deren verwendete Ports
2. Das Skript vergleicht den Zielport jeder eingehenden Verbindung mit der Liste von Hostports
3. Abhängig vom Ergebnis wird eine Meldung in den Log geschrieben

Quelltext 5.4: Abfrage und Abgleich der Ports

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

63  event users_result(ctx: ZeekAgent::Context, data: Columns){
64     local new_entry : count;
65     local connection_port = data$remote_port;
66     new_entry = connection_port;
67     local_ports[new_entry] = data$name;
68  }

77     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. 63).
Loggt für jede Antwort eines DNS-Servers die aufgelöste URL und dazugehörige IP-Adresse (Z. 56).
Im Log vermerkt sind ausgehende Verbindungen, deren Zieladressen vorher nicht durch einen DNS-Server oder die Hosts-Datei aufgelöst wurden (Z. 68).

Quelltext 5.5: Überprüfung der Verbindungsziele eines Endgerätes

```
53 event query_result(ctx: ZeekAgent::Context, data: Columns){
54    local ip_address = to_addr(data$line_content[0]);
55    local host_name = data$line_content[1];
56    resolved_addresses[ip_address] = host_name + " from local hosts";
57 }
58
59 function query_hosts_file(){
60    local str_stmt_hosts = "SELECT columns FROM
          files_columns(\"/etc/hosts\",\"$1:text,$2:text\")";
61    local query_event = query_result;
62    local _schedule = 30 secs;
63    local test_query_join = ZeekAgent::query([$sql_stmt=str_stmt_hosts,
          $event_=query_event, $schedule_=_schedule]);
64 }
65
66 event check_resolve_table(c : connection){
67    local destination_ip = c$id$resp_h;
68    if(destination_ip !in resolved_addresses && destination_ip !in dns_server){
69       local rec: DNStest::Info = [$ts=current_time(), $id=c$id,
             $notice="Connection without Resolve!"];
70       c$dnstest = rec;
71       Log::write(DNStest::LOG, rec);
72    }
73 }
```

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. Untersucht werden sollte, ob und wie vorhandene Kontextinformationen im Bezug auf NSM kategorisiert werden können, welcher Kontext sich zur Anpassung der Arbeitsabläufe eines IDS eignen. Diese sollten zur besseren Übersicht in einer Taxonomie in Beziehung zueinander gesetzt. Ergründet werden sollte, 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 Kontext aus verschiedenen Einzelkategorien der im Kapitel 4 vorgestellten Taxonomie kombiniert, um im jeweiligen Anwendungsfall die IDS-Eigenschaften zu verbessern. Die Nützlichkeit der Kontextinformationen hängt dabei von der Verfügbarkeit und Qualität der Informationen sowie der Komplexität, dass dazu benötigte Skript zu erzeugen ab. Die Qualität und Quantität der Informationen wird daran bemessen, wie hoch der Aufwand war, sie zu beschaffen und in eine Form zu bringen, die eine Verarbeitung zulässt.

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 wird 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 den Grafiken 6.1 und 6.2 bewertet.

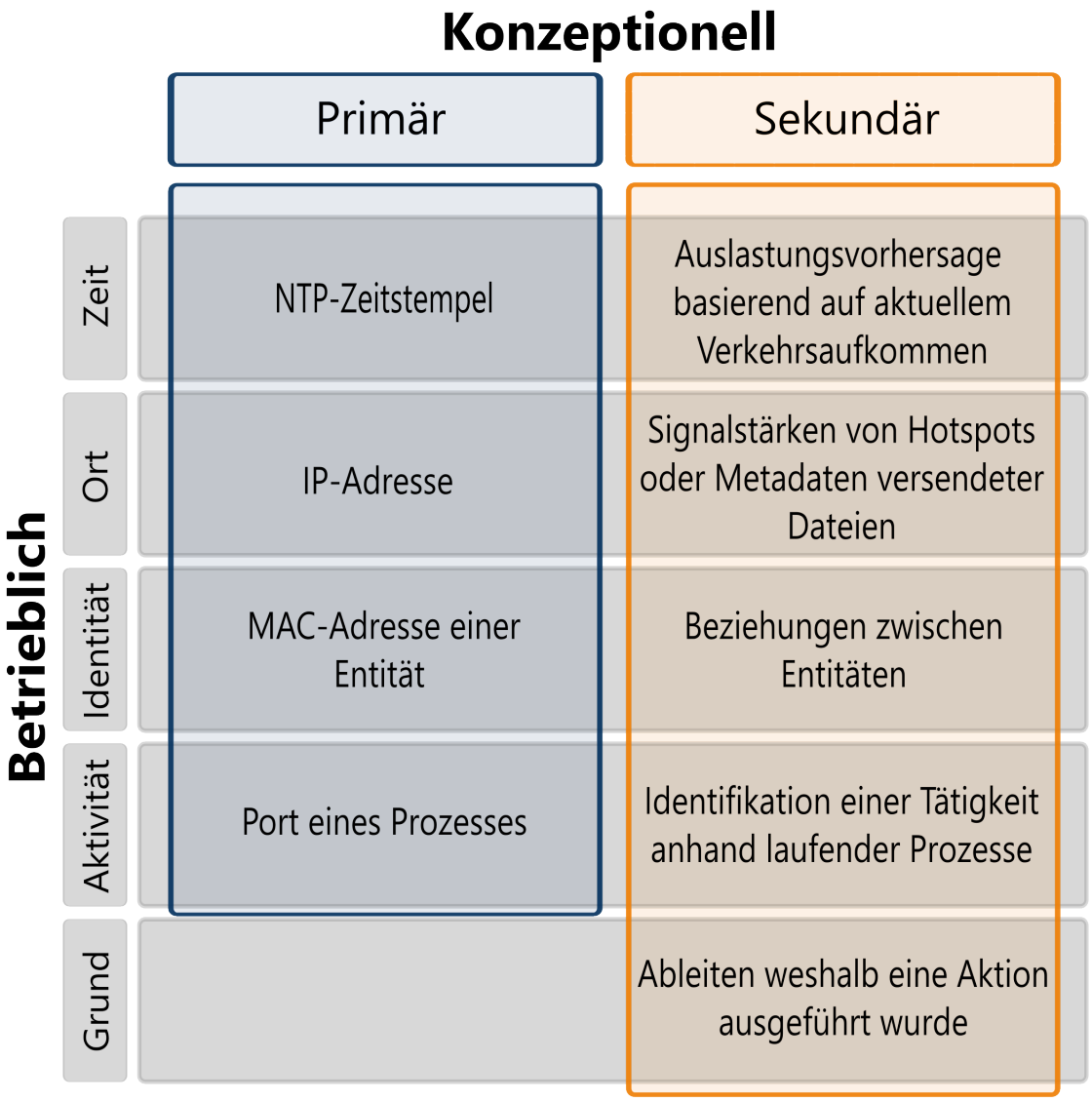

Abbildung 6.1: Evaluation der nutzerzentrierten Kategorisierung

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

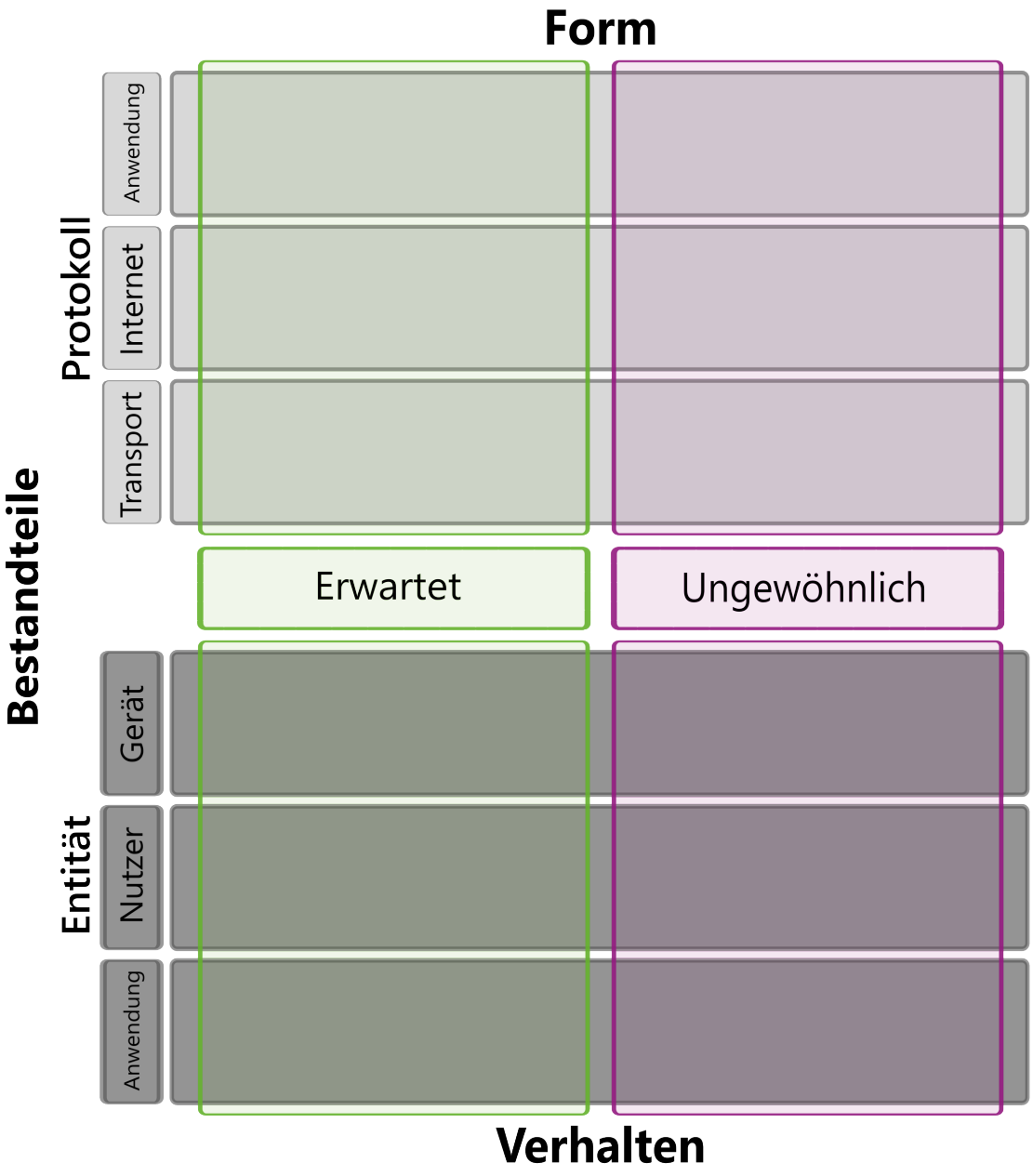

Abbildung 6.2: Evaluation der netzwerkzentrierten Kategorisierung

weniger ein Problem, dass Informationen generell nicht vorhanden sind, sondern lediglich nicht ausreichend schnell zur Verfügung gestellt werden können. Im Fall der verwendeten Ports (5.2) 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 **??**, 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 5.2 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 Kontext sowohl die Erkennung von Angriffen und die Rate an Falsch-positiven verbessert als auch. Die Verwendung zusätzlicher Kontextinformationen im Vergleich zu nicht kontextsensitiven Zugriffskontrollmechanismen, eine geringere

7 Schlussfolgerung und Ausblick

7.1 Dateilose Prozesse

Die vorgestellte Implementation bezieht im besten Fall lediglich Prozesse die in ihrem Lebenszyklus auf die Festplatte schreiben mit ein und versucht dies zu Netzwerkverkehr zuzuordnen. Prozesse die nur im Arbeitsspeicher existieren werden nicht betrachtet sind aber ein zunehmendes Problem.

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 Muling". In: (2007), S. 15.

[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: (2001), S. 4.

[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". In: (2003).

[12] Anita K Jones und Robert S Sielken. "Computer System Intrusion Detection: A Survey". In: *Intrusion Detection* (), S. 25.

[13] A S M Kayes, Jun Han und Alan Colman. "ICAF: A Context-Aware Framework for Access Control". en. In: (2012), S. 8.

[14] Peter Mell u. a. "An overview of issues in testing intrusion detection systems". In: (2003).

[15] Aleksandar Milenkoski u. a. "Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices". en. In: *ACM Computing Surveys* 48.1 (Sep. 2015), S. 1–41. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2808691. (Besucht am 27. 05. 2022).

[16] 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.

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

[18] Bill N Schilit, Norman Adams und Roy Want. "Context-Aware Computing Applications". In: (1994), S. 7.

[19] 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. (Besucht am 15. 09. 2022).

[20] 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.

[21] 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.

[22] 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.

[23] *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).

[24] 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.