Technische Universität München
Fakultät für Informatik

Bachelor's Thesis in Informatik

# Erstellung von Testdatensätzen für Intrusion Detection Systeme

Christian Lübben

# Technische Universität München

## Fakultät für Informatik

Bachelor's Thesis in Informatik

# Erstellung von Testdatensätzen für Intrusion Detection Systeme

# Generating IDS Test Data Sets


| | |
|---|---|
| *Autor* | Christian Lübben |
| *Aufgabensteller* | Prof. Dr.-Ing. Georg Carle |
| *Betreuer* | Nadine Herold, M. Sc., Dipl.-Inf. Stephan-A. Posselt |
| *Datum* | 15.01.2015 |



Informatik VIII
Lehrstuhl für Netzarchitekturen und Netzdienste


Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

Garching b. München, 15.01.2015

Unterschrift

**Zusammenfassung**

Zur Evaluierung von Intrusion Detection Systemen eignen sich Datensätze, die verschiedene Angriffsszenarien beinhalten. Beispiele hierfür sind DARPA 98, KDD 99 sowie NSL-KDD 09. Die Ergebnisse, die durch Evaluation der IDS mit diesen Datensätzen gewonnen werden, lassen wertvolle Schlüsse für die Entwicklung von Intrusion Detection Systemen zu und stellen eine Vergleichbarkeit her. Da die genannten Datensätze bereits einige Jahre alt sind, haben sie den Nachteil, dass die enthaltenen Angriffsszenarien aufgrund der stetigen Weiterentwicklung von Netzwerkangriffen nicht mehr aktuell sind. In dieser Arbeit werden Anforderungen an geeignete Datensätze formuliert und die vorhandenen Datensätze analysiert. Anhand der Erkenntnisse aus der Analyse wird dann ein Konzept vorgestellt, das die Erzeugung einer Vielzahl neuer und individuell konfigurierbarer Datensätze zulässt. Um die gewünschten Funktionen bereitzustellen, werden mehrere Ansätze geprüft, wovon der Vielversprechendste für die Implementierung ausgewählt wird. Die implementierten Teile des Frameworks werden abschließend mithilfe der in der Analyse formulierten Anforderungen evaluiert. Ergebnis der Evaluation ist, dass alle Anforderungen durch das Framework erfüllt werden. Bei der zusätzlichen Performanz-Untersuchung der Logging-Komponente wird zudem festgestellt, dass sich der Ressourcenverbrauch innerhalb eines akzeptablen Bereichs befindet.

**Abstract**

For evaluation of intrusion detection systems (IDS) test data sets are required. These data sets have proven that they offer a lot of useful information for developers. However, the usefulness of the obtained information depends on the quality of the underlying data set. Since the usage of data sets offers a lot of benefits requirements for proper data sets are worked out. By the use of these requirements the existing data sets DARPA 98, KDD 99 and NSL-KDD 09 are analyzed to determine whether they are adequate for use with present IDS. Within this analysis it is pointed out that the existing data sets suffer from several problems. Since they are several years old one of these problems is the lack of scenarios with current relevance whose analysis would provide important data for the development of future security software. Based on the findings of the analysis a new innovative framework for the creation of data sets is presented. Key features are the opportunity of creating a variety of individually configurable data sets and the extendibility to new attacks. After implementation the parts of the new framework are evaluated. As a result of the quality-analysis it is identified that all afore mentioned requirements are met by the framework. Within the additional performance-analysis it is determined that the utilization of resources by logging component stays within an acceptable range.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kapitel 1

# Einleitung

Durch die zunehmende Vernetzung und Größe moderner Rechnernetze wird deren Sicherung gegen Angriffe immer komplexer. Längst sind Menschen mit der Überwachung der einzelnen Rechner und der Erkennung von Angriffen überfordert. Zudem muss auf viele Angriffe sofort reagiert werden. Menschliche Interaktionen bedeuten jedoch immer eine Verzögerung. Zu diesem Zweck wird Software zur automatischen Erkennung von Angriffen entwickelt. Diese Software, die zur automatischen Sicherung der Rechner und Rechnernetze gegen Angriffe eingesetzt wird, wird als Intrusion Detection System (IDS) bezeichnet. Vorteil bei der Nutzung eines IDS ist die Möglichkeit, auch sehr große Rechnernetze in Echtzeit überwachen zu können. Im Optimalfall werden Angriffe sofort erkannt und Gegenmaßnahmen eingeleitet. Somit können der Schaden sowie Kosten, die durch den Angriff entstehen, verringert werden. Darüber hinaus bietet der Einsatz eines IDS die Möglichkeit, weitere Kosten zu senken, indem für die Überwachung des Rechnernetzes weniger Personal benötigt wird. Probleme, die beim Einsatz dieser Systeme auftreten, sind das Nichterkennen von Angriffen sowie die falschpositive Erkennung. Um Aussagen treffen zu können, wie ausgeprägt die Schwächen in diesen beiden Bereichen sind, werden Testdatensätze verwendet. Diese bestehen aus einer Anzahl von zuvor aufgezeichneten Netzwerkdaten, in die Angriffe eingebettet sind. Zur Evaluation werden die Datensätze dem IDS übergeben, um die Angriffe durch dieses erkennen zu lassen. Anhand eines mitgelieferten Schlüssels kann dann geprüft werden, ob die durch das IDS erkannten Angriffe tatsächlich Teil eines Angriffs sind und ob alle tatsächlichen Angriffe durch das IDS erkannt wurden. Da zukünftig die Sicherung von Rechnernetzen nur automatisiert gelingen kann, sind IDS unabdingbar. Somit ist die stetige Weiterentwicklung der IDS notwendig. Um die Qualität der Systeme aussagekräftig vergleichen zu können, werden auch zukünftig Testdatensätze benötigt.

Bekannte Datensätze sind DARPA 98, KDD 99 sowie NSL-KDD 09. Sie enthalten eine Vielzahl von Angriffen aus den Kategorien „denial-of-service“, „probe activity“, „user to root“ und „remote to local“. Die Datensätze sind inzwischen mehrere Jahre alt, so dass aktuelle Entwicklungen im Bereich der Angriffe nicht berücksichtigt werden. Da jedoch auf Grundlage der Datensätze verschiedene IDS verglichen und Aussagen über

deren Qualität getroffen werden, können Probleme auftreten. Es wird versucht, aktuelle IDS mithilfe von Angriffen zu bewerten, die indes nicht mehr relevant sein könnten. Das führt zu der Frage, ob die vorhandenen Datensätze weiterhin für den Einsatz mit aktuellen IDS geeignet sind.

## 1.1 Zielsetzung der Arbeit

Das Ziel dieser Arbeit ist die Erstellung neuer Testdatensätze für Intrusion Detection Systeme. Zu diesem Zweck soll ein Framework erstellt werden, das die Generierung einer beliebigen Anzahl neuer Datensätze zulässt. Die Grundlage für diese Arbeit sind die Datensätze für Intrusion Detection Systeme DARPA 98, KDD 99 sowie NSL-KDD 09. Anhand dieser werden Anforderungen an die neuen Datensätze formuliert, denen die Datensätze des zu erstellenden Frameworks entsprechen sollen. Zudem soll es die Datensätze automatisiert erzeugen. Das bedeutet, dass der Benutzer zu Beginn der Erzeugung lediglich mehrere Parameter zur Definition der Umgebung sowie der Angriffe und des Hintergrundverkehrs festlegt und das Framework daraufhin die Datensätze autonom erzeugen kann.

## 1.2 Gliederung der Arbeit

Die Arbeit gliedert sich in sechs Kapitel. Für die Erzeugung neuer Datensätze werden in Kapitel 2 zunächst Anforderungen an Datensätze formuliert. Im ersten Schritt werden die vorhandenen Datensätze DARPA 98, KDD 99 sowie NSL-KDD 09 vorgestellt und darauf folgend anhand der Anforderungen analysiert. Es wird festgestellt, welche Punkte umgesetzt wurden und welche nur unzureichend oder gar nicht vorhanden sind und somit Anlass zur Kritik geben. Nach der Analyse wird in Kapitel 3 der Entwurf für das Framework erstellt. Dazu werden die Funktionalitäten definiert, die das Framework bieten soll, sowie der Aufbau des Frameworks beschrieben. In Kapitel 4 folgt die Implementierung des Frameworks. Dem Logging, welches die automatisierte Aufzeichnung und Zuordnung von Netzwerkpaketen bezeichnet, wird dort Priorität eingeräumt und es werden mehrere Ansätze zu dessen Umsetzung vorgestellt, von denen einer implementiert wird. Zusätzlich wird die Implementierung zur Erzeugung von Hintergrund- und Angriffsverkehr beschrieben. Die umgesetzte Lösung wird in Kapitel 5 anhand der Anforderungen aus Kapitel 2 evaluiert. Es soll sichergestellt werden, dass die neuen Datensätze fundierte Angaben über die Qualität von IDS zulassen und nicht die selben Kritikpunkte der alten Datensätze enthalten. Zusätzlich wird die Performanz des Logging-Teils gemessen und bewertet. Durch diesen Test soll sichergestellt werden, dass die Implementierung in Echtzeit lauffähig ist und den Betrieb des Rechners nicht einschränkt. Kapitel 6 enthält ein Fazit sowie einen Ausblick auf zukünftige Erweiterungen.

# Kapitel 2

# Analyse

Zur Evaluation von Intrusion Detection Systemen (IDS) werden Datensätze benötigt. Diese Datensätze helfen dabei, Schwächen in den getesteten IDS zu erkennen.
Ein Datensatz besteht im Allgemeinen aus einer, über einen zuvor bestimmten Zeitraum, gesammelten Anzahl von Informationen über Netzwerkaktivitäten in einer festgelegten Testumgebung. Diese Netzwerkaktivitäten bestehen in der Regel aus harmlosem Netzwerkverkehr (Background Traffic), in den Angriffe eingebettet sind. Diese Daten werden dem zu testenden IDS zugeführt und geprüft, ob das IDS Angriffe und harmlosen Hintergrundverkehr unterscheiden kann. Da beim Testen von IDS mit einem Datensatz die gleichen Bedingungen für die IDS geschaffen werden, lassen sich anhand der Ergebnisse Aussagen über die Qualität der getesteten IDS untereinander treffen. Kriterien für die Evaluierung können die Anzahl erkannter Angriffe oder die Rate der Fehlalarme sein. Darüber hinaus werden durch nicht korrekt erkannte Angriffe Schwächen in den IDS aufgedeckt. Dies ist hilfreich für Entwickler, da diese Schwächen nach dem Fund gezielt beheben können.
Zunächst wird in diesem Kapitel ein Überblick über aktuell relevante Angriffe gegeben. Um realistische Daten bei der Anwendung von Datensätzen zu erhalten, sollten die genannten Szenarien beinhaltet werden. Weitere Anforderungen, die ein Datensatz erfüllen sollte, werden im darauffolgenden Abschnitt formuliert.
Existierende Datensätze weisen jedoch einige Schwächen auf. In diesem Kapitel werden drei dieser Datensätze vorgestellt und deren Schwächen anhand der vorher formulierten Anforderungen für Testdatensätze zur IDS Evaluation analysiert.
Die Qualität dieser Datensätze wird immer auch durch deren Nähe zu realen Bedingungen definiert. Somit sind sie auch von der ständigen Weiterentwicklung der Netzwerkbasierenden Angriffe abhängig. Im folgenden Abschnitt werden diese untersucht.

## 2.1 Aktuelle Angriffe

Parallel zu den Fortschritten der Hard- und Softwareindustrie in Rechnernetzen findet die Entwicklung neuer Angriffe statt. Dies können einerseits Weiterentwicklungen

bekannter Angriffsmuster sein oder auch ganz neuen Ansätzen für Angriffsszenarien folgen [1]. Im Folgenden werden Angriffe vorgestellt, die durch aktuelle Erkenntnisse renommierter Institutionen als akute Gefährdungen eingestuft werden.

Ein weit verbreitetes Beispiel für die Weiterentwicklung eines Angriffstyps ist der „distributed denial-of-service“ Angriff (DDoS). Dieser wird im Jahr 2013 vom Bundesamt für Sicherheit in der Informationstechnik (BSI) als eine besonders relevante Gefährdung eingestuft [2]. Auch zukünftig wird diesem Angriffsmuster eine steigende Bedrohung prognostiziert. Intention des DDoS Angriffs ist die Störung der Verfügbarkeit der Zielrechner, durch Überlastung ihrer Ressourcen. Die Weiterentwicklung zum ursprünglichen „denial-of-service“ (DoS) Angriff, dessen Intention identisch ist, umfasst die Nutzung vieler verteilter Rechner für dieses Vorhaben. So muss der Quellrechner bei Flooding-Angriffen beispielsweise nicht mehr über höhere Datenübertragungsraten als das Ziel verfügen, da der Angriff von vielen Hosts mit geringeren Datenübertragungsraten gleichzeitig ausgeführt werden kann [3].

Häufig sind die einzelnen Rechner verteilter Angriffe Teil eines Botnetzes, einer weiteren aktuellen Bedrohung. Botnetze bestehen aus einer Vielzahl einzelner Rechner, die für verschiedene Angriffe wie DDoS Attacken oder den Versand von Spam E-Mails missbraucht werden. In der Regel werden Rechner durch den Download von Schadsoftware Teil eines Botnetzes. In den meisten Fällen ist dem Nutzer nicht bewusst, dass sein Rechner Teil eines Botnetzes ist, da sich die Schadsoftware möglichst unauffällig verhält. Durch einen Botnetz-Operator werden Befehle an die einzelnen Botrechner übermittelt, die somit Teil verteilter Angriffe werden. Der Trend im Bereich der Botnetze neigt zu immer größeren Netzen und immer resistenter werdender Clientsoftware [1].

Die Annahme, dass Angriffe primär vom Internet, also von außerhalb des zu schützenden Netzwerks, stattfinden, ist nicht mehr aktuell. Zwar besteht die höchste Bedrohung auf Rechnernetze durch Angriffe aus dem Internet und Intranet [4], doch aktuelle Angriffe finden auch von innerhalb des zu überwachenden Netzes aus statt. Die zunehmende Nutzung von mobilen Geräten wie Laptops und Smartphones in Rechnernetzen ermöglicht es Angreifern, diese Netze von innen heraus zu infizieren [5]. Eine bloße Beschränkung auf sichere Gateways und gute Firewalls ist heutzutage nicht mehr ausreichend. Somit sind auch Netze ohne Internetanbindung nicht sicher.

Durch die verbreitete Nutzung mobiler Geräte, wie Laptops und Smartphones, birgt eine Vielzahl neuer Gefahren. Bei Verlust oder Diebstahl solcher Geräte könnte ein Unbefugter Zugriff auf wichtige Daten erlangen. Dieses ist ein nicht zu unterschätzendes Risiko. Untersuchungen von Kaspersky zufolge, einem der größten Hersteller von Sicherheitssoftware, sind 2014 bei 26% der befragten Unternehmen Fälle von verlorener oder gestohlener Hardware aufgetreten [5]. Hinzu kommt das Risiko einer Infizierung mit Schadsoftware außerhalb des geschützten Netzes. Vor allem Smartphones sind ein für Angreifer attraktives Ziel. Besonders auf Geräten mit dem Betriebssystem Android wird eine stark zunehmende Anzahl von neuer Schadsoftware festgestellt [1]. Mithilfe dieser infizierten Geräte können sensible Daten gestohlen, Botnetze aufgebaut oder

mittels WLAN andere Netzwerke von innen angegriffen werden [1, 4].

Eine weit verbreitete Methode, laut BSI im Jahr 2014 die zweithäufigste Ursache für interne Angriffe, ist die Nutzung von USB Datenträgern zur Verbreitung von Schadsoftware. Diese können im privaten Umfeld, beispielsweise bei Mitarbeitern zuhause infiziert worden sein oder gezielt in Umlauf gebracht werden. Sie enthalten in der Regel Malware, die bei Ausführung den Rechner infiziert [4]. Neue Ansätze sind jedoch die Manipulation der Firmware von USB Datenträgern, um bei Anschluss dem System vorzugeben, ein anderes Peripheriegerät zu sein. Ein mögliches Szenario ist die Emulation einer Tastatur. Dadurch können mit diesem Datenträger eine Vielzahl verschiedener Aufgaben ausgeführt werden. Denkbar sind das Hinzufügen eines neuen Benutzerkontos oder das Hochladen von Dateien auf einen Server des Angreifers [6].

Die dritthäufigste Bedrohung ist eine Methode, bei der nicht durch direkte Angriffe auf das Rechnernetz Zugang erlangt wird. Beim Social Engineering stehen die Mitarbeiter des Unternehmens im Visier des Angreifers. Ziel ist es, durch die Vorgabe einer vertrauenswürdigen Person, an Informationen zu gelangen. Diese Informationen können beispielsweise Zugangsdaten zu geschützten Bereichen sein, mit denen der Angreifer, als Mitarbeiter getarnt, unbemerkt ins System eindringen kann [4]. Verbreitet ist dieses Verfahren auch beim Online Banking, indem durch Phishing-Angriffe versucht wird, an die Bankdaten der Ziele zu gelangen.

Weitere Möglichkeiten für die Implementierung neuer Angriffe bietet ebenfalls der Einsatz von Virtualisierung. So geben 30% der durch Kaspersky befragten Unternehmen für 2014 an, die Virtualisierung ihrer Infrastruktur voranzutreiben. Jedoch beginnen auch Angreifer damit, Angriffe speziell für virtuelle Umgebungen zu entwickeln, so dass Sicherheitssoftware an die neuen Bedingungen angepasst werden muss. Die Isolation zwischen Host und Gastsystem konnte bereits durch Angriffe überwunden werden [7]. Gerade der Zugriff auf die für die Virtualisierung verantwortlichen Teile des Systems ist kritisch. Wird der Virtual Machine Monitor kompromittiert, werden auch alle virtuellen Maschinen in Mitleidenschaft gezogen. So kann ein Angreifer mit einem erfolgreichen Angriff mehrere Maschinen unter seine Kontrolle bringen.

Eine zunehmende Bedrohung, gerade für große Unternehmen, sind „Advanced Persistent Threats“ (APT). APTs stellen eine Erweiterung normaler Angriffe dar. Sie richten sich nicht wie gewöhnliche Angriffe an eine Vielzahl unbestimmter Ziele, sondern gezielt an bestimmte Netzwerk-Infrastrukturen. APTs sind komplexer als normale Angriffe, da sie aus einer Verkettung verschiedener Angriffe bestehen. Während des Angriffs soll die Tätigkeit des Angreifers so lange wie möglich unerkannt bleiben, um keine Abwehrreaktionen des Ziels hervorzurufen. Ein populäres Beispiel für ein APT ist der im Jahr 2010 entdeckte Stuxnet Computerwurm. Entworfen wurde er, um Zugriff auf ein von Siemens entwickeltes System zur Regulation und Überwachung von Industrieanlagen zu gewähren. Im ersten Schritt wurden dafür zunächst bis dato unbekannte Sicherheitslücken (Zero Day Exploits) im Windows Betriebssystem ausgenutzt. Im nächsten Schritt wurde mittels eines Rootkits die Anwesenheit des Wurms verschleiert, so dass die An-

tivirensoftware des betroffenen Rechners keine Reaktion einleiten konnte. Daraufhin versuchte der Wurm sich im Netzwerk weiter auszubreiten. In der Folge wurden Änderungen an der Software für die Programmierung der Siemens Steuerung vorgenommen, die letztendlich für eine Fehlfunktion dieser sorgen sollten [8].
Die genannten Angriffe stellen eine Auswahl aktueller Bedrohungen für Netzwerk-Infrastrukturen dar. Anhand dieser Auswahl wird deutlich, dass vor allem durch den Einsatz mobiler Geräte eine Vielzahl neuer Angriffsszenarien ermöglicht werden. Auch diese Szenarien müssen von modernen IDS erkannt werden.

## 2.2 Anforderungen

Damit ein Datensatz aussagekräftige und realistische Ergebnisse liefert, sollte er bestimmte Anforderungen erfüllen, die in den folgenden Abschnitten formuliert werden. Die Anforderungen lassen sich unter den Punkten **Dokumentation**, **realistische Annahmen** und **Interaktion zwischen Angriffen und Hintergrundverkehr** zusammenfassen.

#### A1 Dokumentation

Ein zentraler Teil des Datensatzes ist dessen ausführliche Dokumentation. Sie sollte so umfangreich sein, dass eine Replizierung der Experimente jederzeit möglich ist. Aus diesem Grund muss die Dokumentation so detailliert sein, dass ein genaues Verständnis aller Abläufe während der Erzeugung des Datensatzes gewährleistet werden kann. Somit müssen sämtliche Informationen über die Testumgebung enthalten sein. Dies bedeutet, dass neben der Topologie auch Informationen über die verwendete Hardware aller Testrechner sowie die für das Experiment verwendete Software genau spezifiziert werden muss. Zusätzlich zu der verwendeten Software müssen auch alle Programmaufrufe mit den genutzten Parametern angegeben werden. Um am Ende des Experiments ein genaues Verständnis der gesammelten Daten zu ermöglichen, muss jede Datei des fertigen Datensatzes beschrieben werden. Dafür muss der Inhalt der Dateien sowie das genutzte Dateiformat spezifiziert werden.
Wichtig, vor allem für die spätere Evaluation, ist das detaillierte Labeln der einzelnen Pakete. Dies sollte so genau wie möglich sein, um fundierte Aussagen über die Qualität des getesteten IDS treffen zu können. Festzuhaltende Einzelheiten sind Informationen, ob das entsprechende Paket Teil des normalen Netzwerkverkehrs oder eines Angriffs ist. Handelt es sich um einen Angriff, ist dieser zu benennen. Weiterhin sollte festgehalten werden, von welchem Host jedes Paket stammt. Eine Zuordnung, die lediglich durch die Quell-IP-Adresse stattfindet, ist möglich, deckt aber Angriffe nicht ab, die gefälschte IP-Adressen verwenden (Spoofing). Es ist daher nützlich, den Quell-Host explizit anzugeben. Um auch mehrere sich überlagernde Angriffe, die zur gleichen Zeit auf einem Host ausgeführt werden, unterscheiden zu können, muss zusätzlich der zum Paket gehörende Prozess sowie dessen Funktion angegeben werden.

### A2 Realistische Annahmen

Um realistische Ergebnisse zu erzielen, ist die Netzwerktopologie der Testumgebung ein wichtiges Kriterium. Diese sollte so genau wie möglich die Bedingungen einer realen Umgebung eines Produktivnetzes abbilden. Vor allem die Größe des Netzes und die Anzahl der Hosts in der Testumgebung ist ein Faktor, der Auswirkungen auf die Auslastung des IDS hat. Mit steigender Anzahl untereinander kommunizierender Hosts erhöhen sich die Ressourcen-Anforderungen an das IDS, da eine größere Anzahl von Verbindungen durch das IDS überwacht werden muss. Ein weiteres Kriterium, das in Zusammenhang mit der Topologie die Ressourcen der IDS in realistischem Maße auslasten soll, ist die Datenrate in der Testumgebung. Diese ist aktuellen Standards anzupassen, um das durch das IDS zu verarbeitende und zu analysierende Datenvolumen praxisnah zu modellieren. Ziel der beiden Punkte Topologie und Datenrate ist es, möglichst viel Kommunikation zwischen den Rechnern zu erzeugen, die von dem IDS überwacht werden muss. Das IDS soll so in realistischem Maße ausgelastet und gegebenenfalls an Lastgrenzen gebracht werden, um die dadurch entstehenden Folgen messen zu können.
Werden beispielsweise eine zu niedrige Datenraten und gleichzeitig ein sehr kleines Netz simuliert, verstärken sich die Effekte auf das Ergebnis. So kann ein IDS für gut befunden werden, welches einzelne Netzwerkpakete sehr genau und aufwändig untersucht. Somit werden eine gute Erkennung von Angriffen sowie eine geringe Fehlalarmquote ermöglicht. Wird dieses IDS jedoch in der Praxis unter realen Bedingungen eingesetzt, kann es durch das weit größere zu verarbeitende Datenvolumen und die vielen Teilnehmer schnell überlastet sein. Die Anzahl der zu verarbeitenden Pakete pro Zeiteinheit steigt und lässt eine aufwändige Verarbeitung durch das IDS aufgrund zu hoher Ressourcenauslastung nicht mehr zu. Dies hat zur Folge, dass das IDS nicht mehr uneingeschränkt funktionsfähig ist, was somit negative Auswirkungen auf die zuverlässige Erkennung von Angriffen hat.
Neben den Anforderungen an die Testumgebung sind ebenfalls Anforderungen an die Auswahl der Angriffe geknüpft. Für diese gilt zum ersten, dass sie im Mittel ein erhöhtes Maß an Schadpotential besitzen sollten. Es ist wenig zielführend, IDS anhand von Angriffen zu evaluieren, aus denen kein nennenswerter Schaden resultieren würde. Des Weiteren sollten die Angriffe, wie die in 2.1 genannten Angriffe, aktuelle Relevanz besitzen, also auch real auftreten. An diesem Punkt muss jedoch eine Abschätzung getroffen werden, zwischen Standardangriffen, die extrem häufig auftreten und daher schon eine gewisse Relevanz besitzen, und selteneren Angriffen, die kein allzu bekanntes Schema verwenden, jedoch ebenfalls großen Schaden anrichten können. Es sollte auch vermieden werden, ausschließlich Angriffe eines Typs wie beispielsweise „denial-of-service“ (DoS) auszuwählen, um eine Vielfalt, wie sie auch real auftritt, zu gewährleisten. Neben der Auswahl von mehreren Angriffstypen sollten die Angriffe im Netz nicht immer von den selben Hosts ausgehen, um zu vermeiden, dass der Verkehr eines bestimmten Hosts von vornherein als Angriff klassifiziert werden kann, also eine Erkennung nur aufgrund der Absender IP möglich ist.

Um die Angriffe möglichst realitätsgetreu in den Netzwerkverkehr einzubetten, wird ein realistischer Hintergrundverkehr benötigt. Zum Erzeugen von Hintergrundverkehr gibt es verschiedene Herangehensweisen. Es besteht die Möglichkeit, einen Traffic-Generator zu verwenden, der zuvor generierte Pakete in das Netz einspeist. Demgegenüber besteht die Möglichkeit, mithilfe der Clients im Netz, den Verkehr real zu erzeugen, indem jeder Client mit anderen Clients im Netz oder dem Internet interagiert. Als realistischer lässt sich die direkte Erzeugung von Hintergrundverkehr herausstellen, da die Pakete und deren Interaktion real erzeugt werden und nicht versucht wird, die Realität künstlich abzubilden. Dabei muss darauf geachtet werden, dass der Verkehr nicht zu einfach ist. Es sollten zum einen mehrere Protokolle zum Einsatz kommen und nicht beispielsweise ausschließlich *HTTP*-Verkehr. Zum anderen sollten nicht nur redundante Aufrufe, wie das ständige neu laden einer Website oder das wiederholte Senden einer Datei über das Netzwerk stattfinden.

#### A3 Interaktion zwischen Angriffen und Hintergrundverkehr

Obwohl die Interaktion zwischen Angriffen und Hintergrundverkehr der Abbildung realistischer Annahmen dient, wurde dieser Punkt einzeln hervorgehoben, um die Wichtigkeit dieses Aspekts zu unterstreichen. Auch bei Erfüllung aller weiteren Anforderungen hat das Fehlen der Interaktion maßgebliche Folgen für die Qualität der erzeugten Datensätze.

Um ein realistisches Verhalten des Netzwerks auf einen Angriff zu ermöglichen, muss der Hintergrundverkehr auf die Angriffe reagieren. Dies bedeutet beispielsweise, dass ein Client der von einem DoS Angriff betroffen und überlastet ist, nicht wie zuvor weiterhin am Netzwerkverkehr teilnimmt, sondern im äußersten Fall keinen Verkehr mehr erzeugt. Das Fehlen dieser Auswirkungen kann vor allem beim Training von anomaliebasierten IDS zu Fehlinterpretationen führen. Diese Interaktion kann durch die reale Erzeugung von Netzwerkverkehr während der Aufzeichnung des Datensatzes modelliert werden. Traffic-Generatoren erlauben die Modellierung solche Effekte in der Regel nicht. Es werden unabhängig davon, ob ein Client angegriffen wird, weiterhin Pakete in dessen Namen versendet. Die Zusammenführung von Angriffs- und Hintergrundverkehr erfolgt durch eine bloße Überlagerung der Angriffs- und Hintergrundverkehrsdaten. Somit sind keine Auswirkungen des Angriffs auf den Netzwerkverkehr im System sichtbar.

## 2.3 Bestehende Datensätze

Im Folgenden werden drei bestehende Datensätze vorgestellt. Die Auswahl beinhaltet die Datensätze, die bis dato die Referenz für IDS Testdatensätze darstellen.

### 2.3.1 Darpa 98

Die erste Version des DARPA Intrusion Detection Evaluation Data Set wurde 1998 durch das Lincoln Laboratory des MIT veröffentlicht. Finanziert wurde das Projekt durch die Defense Advanced Research Projects Agency (DARPA) des Verteidigungsministeriums der Vereinigten Staaten [13].
Der Datensatz beschreibt eine fiktive Air Force Base, deren Topologie (Abb. 2.1) der einer typischen Air Force Basis nachempfunden wurde. Diese fiktive Basis trägt den Namen Eyrie AFB und besteht aus vier realen Maschinen, sowie einer Maschine, die als Traffic-Generator dient [10]. Als Betriebssysteme auf den realen Maschinen werden SunOS 4.1.4 (Zeno), Linux Redhat 4.0 mit Kernel 2.0.27 (Marx) und Solaris 2.5.1 (Locke, Pascal) eingesetzt. Auf dem Traffic-Generator (Hobbes) kommt Linux Redhat 5.0 mit einem modifizierten Kernel 2.0.32 zum Einsatz. Zusätzlich zu den Realen werden weitere virtuelle Maschinen (VM) eingesetzt. Unter diesen virtuellen Maschinen befinden sich zehn Linux Rechner (Linux 1-10), Rechner mit verschiedenen Windows Versionen (3.1, 95, NT 4.0) und MacOS Betriebssystem (pc 1-9) [10, 13]. Da neben der Aussage, dass auf Hobbes mehrere VMs laufen, keine genaueren Angaben gemacht werden, auf welchem Host welche VMs laufen, wird an dieser Stelle die Annahme getroffen, dass alle weiteren Maschinen, die im Diagramm nicht verzeichnet sind, auf Hobbes gehostet werden. Zusätzlich wird in [12] die Aussage getroffen, dass mittels des modifizierten Kernels von Hobbes die Illusion von virtuellen Maschinen erstellt werden kann. Dies soll dadurch gelingen, dass Prozesse bei Erstellung eine IP Adresse spezifizieren können, die als Quelladresse für den gesamten Verkehr dient, den der Prozess erzeugt. Dies lässt die Vermutung zu, dass die virtuellen Linux Maschinen lediglich simuliert werden.
Die Verbindung mit einem äußeren Netz wird über einen Cisco 2514 Router hergestellt. In diesem äußeren Netz, welches das Internet simulieren soll, befinden sich ebenfalls ein Traffic-Generator und ein Webserver, der verschiedene Domains bereitstellt. Zusätzlich befindet sich ein Rechner mit SunOS 5.6 (Solomon) in diesem Netz, der zur Aufzeichnung des Netzwerkverkehrs für den Datensatz dient [13].
Der Datensatz besteht zum einen aus der Aufzeichnung des Netzwerkverkehrs, der mithilfe von *Tcpdump* erstellt und im *.pcap*-Format vorliegt. Dies ist ein verbreitetes Dateiformat, das für die Aufzeichnung von Netzwerkpaketen am Netzwerkinterface des Hosts mithilfe der *pcap*-Programmierschnittstelle verwendet wird [14]. Der Name leitet sich von „packet capture“ (Paket erfassen) ab. Die Aufzeichnung findet außerhalb der Basis auf dem Rechner Solomon statt und erfasst den Verkehr in die Basis hinein und aus ihr heraus. Zum anderen beinhaltet der Datensatz eine Logdatei, die auf Pascal mithilfe des „Solaris Basic Security Module“ (BSM) erfasst wurde. Diese umfasst detaillierte Informationen über den Aufruf von „system calls“ auf dem Host. Darüber hinaus beinhaltet der Datensatz Dateisystemauszüge der drei Angriffsziele Marx, Zeno und Pascal. Mithilfe dieser Daten können Änderungen beispielsweise an Systemprogrammen festgestellt werden, die nicht von normalen Benutzern vorgenommen werden können. So kann festgestellt werden, ob das betreffende System kompromittiert wurde [12].

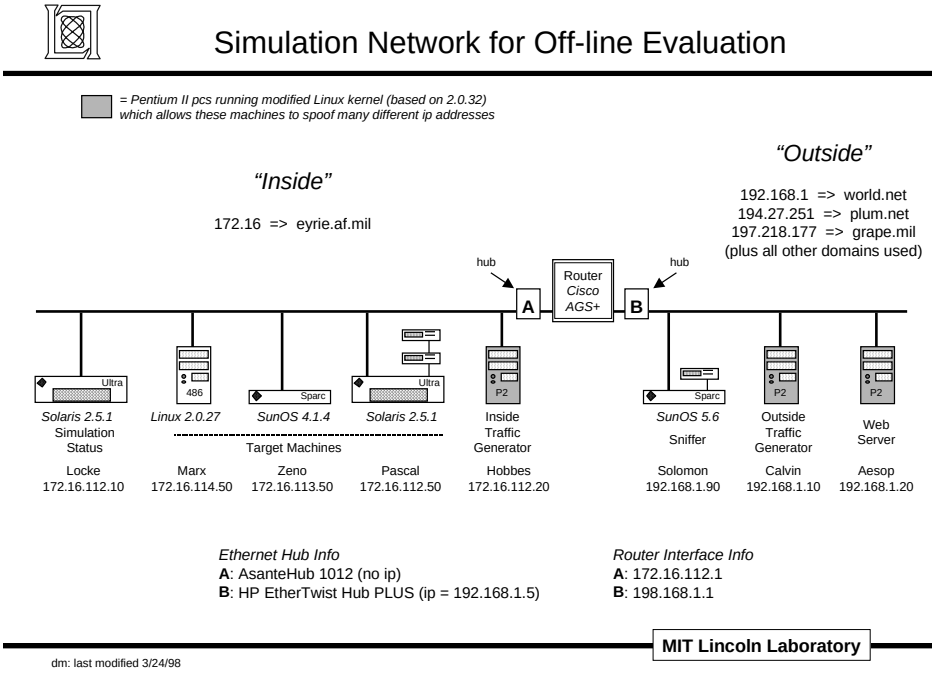


Abbildung 2.1: Topologie Darpa 98 [13]

Die erfassten Daten gliedern sich zudem in Trainingsdaten sowie Testdaten. Die Trainingsdaten umfassen sieben Wochen mit jeweils fünf Aufzeichnungstagen. Pro Tag werden jeweils ungefähr 22 Stunden aufgezeichnet. Mithilfe dieser Trainingsdaten können die IDS auf die Evaluation vorbereitet werden. Bei signaturbasierten IDS können die Regeln geprüft werden und weitere Justierungen erfolgen. Dies ist jedoch auch ein Kritikpunkt, da die Daten keineswegs auf realistisches Verhalten verifiziert wurden. Es besteht die Gefahr, IDS dahingehend zu optimieren, bei der anschließenden Evaluation gute Ergebnisse zu erzielen, jedoch für den realen Einsatz völlig unpassend zu sein. Für die Vorbereitung von anomaliebasierten IDS besteht ein Problem darin, dass auch die Trainingsdaten Angriffe enthalten und somit keine Basis für den „normalen“ (angriffsfreien) Ablauf im System bieten [10].

Die eigentlichen Testdaten zur Evaluation umfassen einen Zeitraum von zwei Wochen zu je fünf Aufzeichnungstagen. Auch hier beträgt die Länge der Aufzeichnungen jeweils ungefähr 22 Stunden pro Tag [10].

Zu den gesammelten Testdaten werden Schlüssel (Truth Lists) mitgeliefert, mithilfe derer eine Zuordnung der gesammelten Pakete in harmlosen Netzwerkverkehr und in Angriffspakete möglich ist. Diese liegen als *.lllist*-Dateien im Attribute-Relation File

Format (ARFF) vor. In diesem Dateiformat werden zuvor definierte Spaltenattribute zeilenweise hintereinander aufgelistet. Diese sind unter anderem die Start- und Zieladressen, Zeitstempel und genutzten Protokolle. Die Trennung der Attribute erfolgt durch Leerzeichen [13, 15].
Die Dokumentation des Datensatzes umfasst ein komprimiertes *.tar*-Archiv mit einer *readme*-Datei, in der die einzelnen Dateien der Dokumentation aufgelistet und beschrieben werden. Die weiterführende Datei *README.formats* befasst sich mit dem *ARFF*-Dateiformat, in dem die Schlüssel vorliegen. Darin werden die Zeilen und Spalteneinträge detailliert erläutert. In zwei weiteren *readme*-Dateien (*.bsm, .tcpdump*) werden Angaben über die beiden Aufzeichnungsprogramme *Tcpdump* und BSM gegeben. Für *Tcpdump* werden die Aufrufparameter und für BSM die Konfiguration und bekannte Programmfehler (Bugs) genannt. Neben den *readme*-Dateien beinhaltet die Dokumentation zwei Beispieldateien im *ARFF*-Format. Ein Netzwerk Diagramm (Abb. 2.1) ist in mehreren Formaten vorhanden(*.gif, .ps, .ppt*). Des Weiteren wird eine Liste mit den im Netzwerk befindlichen Hosts und deren Name, IP-Adresse und Betriebssystem mitgeliefert. Weiterhin ist eine Auflistung der Angriffe mit Zeitpunkt, Angriffsname, Quell-IP und Ziel, sowie einer kurzen Beschreibung der Angriffe und eine Liste mit anormalen Aktivitäten zum Testen von anomaliebasierten IDS beinhaltet [13].
Unter der genutzten angreiferorientierten Taxonomie [16] werden die verwendeten Angriffe in vier Gruppen zusammengefasst. Im gesamten Datensatz, Trainings- und Testdaten, kommen 38 verschiedene Angriffe vor [13]. Elf dieser Angriffe fallen in die Gruppe „denial-of-service“, also Angriffe mit der Absicht, das Ziel lahmzulegen, so dass dieses seinen Aufgaben nicht nachkommen kann. Die Gruppe der „remote-to-user“ Angriffe umfasst 14 verschiedene Angriffe. In diese Kategorie fallen diejenigen, bei denen ein Angreifer im Netzwerk versucht auf einen anderen Rechner zuzugreifen, auf dem er kein Benutzerkonto hat, und dort Benutzerrechte eines der Nutzer des Zielrechners zu erlangen [12]. Eine weitere Kategorie, die sieben Angriffe umfasst, sind die „user-to-root“ Angriffe. Ausgangssituation ist ein Angreifer, der bereits Benutzerrechte auf dem Zielrechner besitzt und mit dem „user-to-root“ Angriff versucht, Administratorrechte (Rootrechte) zu erlangen. Die übrigen sechs Angriffe gehören der Gruppe der „surveillance“ und „probe“ Angriffe an. Das Bestreben dieser Angriffe ist, Informationen über ein Netzwerk oder einzelne Hosts in diesem Netzwerk zu sammeln und gegebenenfalls Schwachstellen offen zu legen [12].
Zu den anormalen Aktivitäten zählen Nutzer, die sich zu unüblichen Zeiten (Nachts) einloggen, von einer anderen Quelle als gewöhnlich auf andere Rechner verbinden, neue Befehle ausführen oder ihre Rollen wechseln (z.B. Sekretär zu Manager) [13].

### Kritik

*A1 Dokumentation*—Die Dokumentation des Darpa 98 Datensatzes umfasst eine Vielzahl an Informationen zu den genutzten Aufzeichnungswerkzeugen sowie den einzelnen Hosts. Trotzdem bestehen jedoch Unklarheiten. Ein wichtiger Teil, der aber nur ober-

flächlich beschrieben wird, ist die Erzeugung des Hintergrundverkehrs. Es wird erklärt, welche Annahmen zugrunde liegen. Der typische Netzwerkverkehr einer Air Force Basis soll modelliert werden. Die Rede ist von einer fiktiven Basis, für die der Hintergrundverkehr kein Mitschnitt eines Produktivnetzes ist, sondern künstlich erzeugt wird. Daher sollte verifiziert werden, ob der für die Erzeugung des Datensatzes genutzte Hintergrundverkehr diesen Vorgaben entspricht. Behauptet wird zwar, dass für die Modellierung der Verkehr einer echten Basis über einen Zeitraum beobachtet wurde, nachvollziehbar und Teil der Dokumentation ist dies nicht [10]. Die Software, die zur Verkehrserzeugung genutzt wird, wird ebenfalls nicht genannt, was die Replizierung zusätzlich erschwert.
Das Labeling der einzelnen Pakete weist ebenfalls Schwächen auf. Neben der Angabe, ob ein Paket bösartig ist und, wenn ja, Teil welches Angriffs es ist, sind keine zusätzlichen Informationen gegeben. Wünschenswert wäre an dieser Stelle eine explizite Angabe des Quell-Hosts, da diese Information durch Spoofing nicht immer aus den IP Adressen rekonstruierbar ist. Durch die fehlende Prozessangabe können darüber hinaus mehrere gleichzeitige Angriffe von einem Host nicht immer zweifelsfrei unterschieden werden.

*A2 Realistische Annahmen*—Auch unter dem Aspekt der realistischen Annahmen weist der Darpa 98 Datensatz Schwächen auf. Die Topologie des Rechnernetzes war bereits zum Zeitpunkt der Erstellung mit vier physikalischen Rechnern sehr klein gewählt [10]. Es wurde versucht diese Tatsache zu kompensieren, indem mithilfe des Traffic-Generators Pakete mit unterschiedlicher Ziel und Absender IP erzeugt wurden. Somit sollte eine Illusion weiterer virtueller Maschinen [12] erzeugt werden. Die Anzahl, der durch den Traffic-Generator gehosteten Rechner, beträgt 20, so dass sich im Netz der Eyrie AFB eine Gesamtanzahl von 25 Rechnern [13] befindet. Diese Anzahl von Rechnern ist jedoch bereits im Jahr 2000 schon als zu wenig kritisiert worden, [10] und mit dem Fortschreiten der technologischen Entwicklung und immer größer werdenden Netzen ist diese Anzahl an heutigen Maßstäben gemessen noch kritischer zu sehen.
Zusätzlich zu dem kleinen Netz sind auch die Datenraten relativ gering. Sie betragen an verschiedenen Tagen durchschnittlich zwischen 8.8 und 51.4 kbit/s [10]. Diese Werte waren ebenfalls für damalige Verhältnisse, in denen bereits Datenraten von mehreren Megabit pro Sekunde möglich waren, nicht hoch genug, um ein IDS an realistische Auslastungen zu bringen.
Zu den Angriffen ist festzuhalten, dass deren Anzahl von 38 unterschiedlichen Angriffen eine ausreichende Menge darstellt. Es sind aber auch in diesem Bereich Schwächen zu finden. Zu diesen zählt, dass Pakete, die Teil eines Angriffs sind, bestimmte Merkmale aufweisen. So besitzen ausschließlich Angriffspakete „time to live"(TTL) Werte von 126 oder 253 [17, 18]. Hinzu kommt die Tatsache, dass lediglich drei der Hosts Ziele interaktiver Angriffe sind. Folglich reicht es bei der Evaluation, lediglich die drei Angriffsziele durch das IDS zu überwachen. Dadurch ist es möglich, das Ergebnis in positiver Richtung zu manipulieren. Erreicht wird dies, da einerseits weniger Verkehr

zu analysieren ist und andererseits, dank Ausschluss eines Großteils von angriffsfreiem Verkehr, weniger falsch-positive Erkennungen auftreten können [10]. Des Weiteren ist es durch die Merkmale der Angriffspakete leichter, diese zu erkennen. Es können beispielsweise Signaturen für bestimmte TTL Werte angelegt werden. Die Annahme, die bei diesem Datensatz getroffen wird, ist jene, dass Angriffe lediglich von außerhalb der Basis stattfinden. Andere Angriffe würden durch die Platzierung des Aufzeichnungsortes außerhalb der Basis nicht erkannt werden. Szenarien, wie die in 2.1 beschriebenen, in denen ein Angriff von innerhalb der Basis beispielsweise durch die Nutzung eines infizierten USB Datenträgers oder Einbindung eines infizierten Mobilgerätes, wie Smartphone oder Laptop, können nicht modelliert werden.
Der Hintergrundverkehr des Datensatzes umfasst eine Vielzahl unterschiedlicher Protokolle, wie beispielsweise *HTTP*, *SMTP* oder *Telnet*. Dies genügt durchaus den Anforderungen, jedoch ist der gesamte Verkehr künstlich erzeugt. Dies führt dazu, dass die „Internet background noise“ (IBN), auch bekannt als „Internet background radiation“, fehlt [18]. Damit ist gemeint, dass nicht produktiver Netzwerkverkehr auftritt. Dies können Pakete sein, die inkorrekte Header besitzen und beispielsweise aus Fehlkonfigurationen im Netzwerk resultieren. Von IDS können diese Pakete als Angriff fehlinterpretiert werden, womit sich auch die Fehlalarmquote ändert.

*A3 Interaktion zwischen Angriffen und Hintergrundverkehr*—Neben den inhaltlichen Schwächen des Hintergrundverkehrs tritt auch in technischer Hinsicht ein gravierender Mangel auf. Dies betrifft die Nutzung eines Traffic-Generators. Durch das Einspielen von Netzwerkverkehr mittels eines Traffic-Generators ist die Interaktion zwischen diesem und den Angriffen unmöglich. Die Addition von Hintergrund- und Angriffsverkehr mag bei einigen Angriffstypen möglicherweise keine Änderung zu real erzeugtem Verkehr haben. Bei Angriffen wie DoS ist jedoch sehr wohl eine Interaktion zu beobachten. Von diesem Angriffstyp befinden sich immerhin elf verschiedene Varianten im Angriffsspektrum. Das Problem, welches aus fehlender Interaktion resultiert, ist, dass IDS die Situationen fehlinterpretieren können oder falsche Verhaltensweisen erlernen. Ein Beispiel für eine Fehlinterpretation bei DoS Angriffen ist, dass der Angriff als harmlos eingestuft und keine Reaktion darauf eingeleitet wird, da das Ziel nicht beeinträchtigt zu sein scheint. Geht es um das Erlernen von Verhaltensweisen, so erlernt das IDS den Angriff lediglich an der Häufung von Paketen und nicht an der Überlastsituation der Ziels zu erkennen.

### 2.3.2 KDD 99

Der KDD 99 Datensatz erschien 1999, ein Jahr nach dem Darpa 98 Datensatz. Er wurde im Rahmen der fünften Konferenz für „Knowledge Discovery and Data Mining“ (KDD-99) veröffentlicht, aus der ebenfalls der Name des Datensatzes hervorging.
Der KDD 99 Datensatz leitet sich vom Darpa 98 Datensatz ab, da keine neuen Daten aufgezeichnet, sondern die des Darpa 98 überarbeitet wurden. Somit bleiben die umge-

bungsspezifischen Parameter identisch zu denen des Darpa 98 Datensatzes. Überarbeitet wurden die Daten von Prof. Salvatore J. Stolfo von der Columbia University und Prof. Wenke Lee von der North Carolina State University [19].

Die Änderungen gegenüber dem Darpa 98 bestehen darin, dass der Datensatz keine einzelnen Pakete mehr beinhaltet. Die Paketdaten des Darpa 98 wurden zu Verbindungen zusammengefasst. Der gesamte Datensatz umfasst ungefähr sieben Millionen Verbindungen, von denen etwa fünf Millionen Einträge zu den über sieben Wochen aufgezeichneten Trainingsdaten und etwa zwei Millionen Einträge zu den über zwei Wochen aufgezeichneten Testdaten des Darpa 98 Datensatzes gehören. Jede dieser Verbindungen umfasst etwa 100 B [20].

Bei der Zusammenfassung der Pakete zu Verbindungen wurden Vektoren erstellt, die jede Verbindung charakterisieren. Die Zusammenfassung zu Vektoren wurde vorgenommen, um den Datensatz nutzbar für Lernalgorithmen zu machen. Der Datensatz mit einzelnen Paketen besitzt eine sehr hohe Komplexität. Die Gruppierung nach ausgewählten Merkmalen ermöglicht es Maschinen-Lernalgorithmen, Muster für die Erkennung von Angriffen zu identifizieren. Im Fall des KDD 99 Datensatzes bestehen diese Vektoren aus 41 Merkmalen [20]. Zu diesen Merkmalen zählen Basisangaben, wie die Dauer der Verbindung, das verwendete Protokoll (*TCP*, *UDP* oder *ICMP*), der verwendete Netzwerkdienst (z.B. *HTTP*) und die Anzahl der versendeten Bytes vom Quell- sowie Zielrechner. Weitere Merkmale, die sich zur Erkennung anormaler Aktivitäten eignen, sind beispielsweise die Anzahl gescheiterter Logins oder Versuche, Befehle mit Administratorrechten auszuführen. [21].

Die veröffentlichten Daten umfassen neben einer Auflistung und Beschreibung der 41 Merkmale eine Liste von Angriffen in den Testdaten, die gelabelten Trainingsdaten sowie die nicht gelabelten Testdaten. Darüber hinaus wurden auch Teildatensätze, die jeweils zehn Prozent der ursprünglichen Test- und Trainingseinträge enthalten, veröffentlicht [21].

Die bereitgestellten Daten zu den Verbindungen liegen im *ARFF*-Format vor. Zusätzlich zu den reinen Daten wird auch bei diesem Datensatz ein Schlüssel mitgeliefert, mit dem die Zuordnung von Verbindungen zu Angriffen und Hintergrundverkehr möglich ist. Die Daten dieses Schlüssels liegen ebenfalls im *ARFF*-Format vor. Die Zeilen entsprechen jeweils einem Verbindungsvektor, die 41 Merkmale bilden dessen Spaltenattribute. Die Schlüssel enthalten ein zusätzliches Attribut, das die Zuordnung zum Hintergrundverkehr oder den zugehörigen Angriff angibt.

Die Dokumentation des Datensatzes beschränkt sich auf eine kurze Einführung in die Thematik der Intrusion Detection sowie einen Überblick über den Darpa 98 Datensatz. Zusätzlich wird auf die Gruppierung der Pakete zu Verbindungen eingegangen und eine Liste der Merkmale mit Name, Beschreibung und Datentyp bereitgestellt. Die Informationen zur Topologie des Rechnernetzes sowie zur Aufzeichnung der Daten sind auf wenige Aspekte begrenzt [21].

### Kritik

*A1 Dokumentation*—Die Dokumentation des Datensatzes lässt Angaben über den Clustering-Algorithmus vermissen. Es werden also keinerlei Angaben gemacht, wie die Pakete zu den einzelnen Vektoren zusammengefasst werden. Weiterhin werden keine Aussagen, die die Wahl der einzelnen Attribute begründen, gemacht. Für Informationen zur Topologie oder weiteren Details über die Aufzeichnung des Datensatzes muss die Dokumentation des Darpa 98 konsultiert werden, da dies beim KDD 99 Datensatz nur sehr oberflächlich beschrieben wird.
Das Labeling der Verbindungen umfasst, wie beim Darpa 98 Datensatz, nur Angaben, ob die Verbindung Teil eines Angriffs ist und wenn ja, den zugehörigen Namen des Angriffs. Detailliertere Einzelheiten über Host und Prozess fehlen.

*A2 Realistische Annahmen*—Der KDD 99 Datensatz verwendet ausschließlich Daten des Darpa 98 Datensatzes. Da diese Daten lediglich für die Verwendung mit Lernalgorithmen in ein anderes Format konvertiert wurden, ergeben sich keinerlei Änderungen oder Verbesserungen hinsichtlich der realistischen Annahmen im Gegensatz zum Darpa 98 Datensatz. Folglich ist das Rechnernetz sehr klein, die Datenraten sind gering, interaktive Angriffe auf einen kleinen Teil der Rechner beschränkt und Angriffe finden nur von außerhalb der Basis statt.
Ein neues Problem ergibt sich jedoch aus dem Zusammenfassen der Pakete. Wie bereits bei der Dokumentation angemerkt wurde, fehlt eine Begründung für die Auswahl der Merkmale in den Vektoren. Diese Auswahl ist jedoch ein wichtiger Aspekt, da die Zusammenfassung der Pakete immer Verlustbehaftet ist, muss die Auswahl der Merkmale die für die Erkennung von Angriffen relevanten Informationen beinhalten. So kommen Adetunmbi et. al in [22] zu dem Schluss, dass nicht alle der 41 gewählten Merkmale für die Erkennung von Angriffen relevant sind. Zudem ist unklar, ob alle relevanten Merkmale vorhanden sind.
Ein weiteres Problem, das durch das Zusammenfassen zu Verbindungen entsteht, ist eine Vielzahl redundanter Verbindungsvektoren. So bestehen die Trainingsdaten zu 78 Prozent und die Testdaten zu 75 Prozent aus redundanten Einträgen [17]. Dies beeinflusst die Bewertung der getesteten IDS. Die Lernalgorithmen werden auf häufig auftretende Einträge fokussiert. Jedoch können auch selten auftretende Einträge extrem schädliche Angriffe beinhalten. Bei der Evaluation nach korrekt zugeordneten Einträgen wird den häufig vorkommenden Verbindungen eine höhere Priorität eingeräumt. Die Auswertung dieser Verbindungen hat folglich eine große Auswirkung auf die Anzahl der insgesamt korrekt ausgewerteten Verbindungen und verschiebt diese zugunsten der IDS, welche die redundanten Einträge korrekt zuordnen [17]. Eine Aussage über die Qualität, anhand korrekt zugeordneter Verbindungen, lässt sich somit schwer treffen, da nicht berücksichtigt wird, welche Konsequenzen durch die falsch zugeordneten Verbindungen zu erwarten wären.
Untersuchungen mit verschieden IDS haben zudem gezeigt, dass ein sehr hoher An-

teil der Verbindungen von den meisten IDS korrekt zugeordnet wurden. Durch diese hohen Erkennungsraten liegen die Evaluationsergebnisse der einzelnen IDS sehr nah beieinander. Bei der Bewertung, anhand korrekt zugeordneten Verbindungen, sollte ein anspruchsvoller Datensatz zum Einsatz kommen, in dem IDS durchschnittlich eine geringere Erkennungsrate aufweisen, so dass anhand der Fehler eine aussagekräftige Unterscheidung der IDS möglich ist [17].

*A3 Interaktion zwischen Angriffen und Hintergrundverkehr*—Zu diesem Punkt gilt ebenfalls die Aussage, dass durch Verwendung der Daten des Darpa 98 Datensatzes dessen Schwächen in diesem Punkt übernommen werden. Es findet also ebenfalls keine Interaktion zwischen Angriffen und Hintergrundverkehr statt. Folglich können auch beim KDD 99 Datensatz Angriffe, wie ein DoS, nicht realistisch modelliert werden, da das Ziel keine Anzeichen von Überlastung zeigt.

### 2.3.3 NSL-KDD 09

Der NSL-KDD 09 Datensatz aus dem Jahr 2009 basiert auf dem KDD 99 Datensatz. Tavallaee et al. stellen in [17] die Änderungen zum KDD 99 Datensatz vor.
Zur Erstellung ihres Datensatzes analysieren sie den KDD 99 Datensatz und kommen zu dem Schluss, dass dieser einen großen Anteil redundanter Informationen enthält. Diese werden als nicht nützlich erachtet. Begründung für die Einstufung als nicht nützlich ist die Tatsache, dass ein IDS mit wiederholt gleichen Eingaben voraussichtlich auch immer gleich reagiert und so ein und das selbe Experiment nur mehrfach durchgeführt wird. Bei Bewertung nach korrekter Zuordnung zu gutartigem oder bösartigem Netzwerkverkehr wird darüber hinaus das Ergebnis für IDS, welche die häufig wiederholt auftretenden Einträge zuverlässig korrekt einstufen, besser. Somit ist die erste Änderung die vorgenommen wird, die Streichung redundanter Einträge. Der Anteil dieser Einträge wird auf etwa 75 Prozent des Datensatzes beziffert [17].
Daraufhin werden 21 IDS ausgewählt, welche mit den Test- und Trainingsdaten des KDD 99 Datensatzes evaluiert werden. Bei der Auswertung der Ergebnisse wird der Schlüssel „successfulPrediction“ eingeführt. Für jeden Verbindungsvektor wird der Wert unter diesem Schlüssel mit Null initialisiert. Daraufhin wird für jedes IDS geprüft, ob dieses eine korrekte Vorhersage bezüglich Angriff oder Nichtangriff getroffen hat. Ist die Aussage korrekt, wird der Wert des Eintrages um eins inkrementiert. Ein Eintrag, der von allen IDS richtig zugeordnet wurde, besitzt folglich den Wert 21. Anschließend werden fünf Schwierigkeitsgruppen gebildet. Als Resultat dieses Vorgehens wird festgehalten, dass Einträge, die nur von 0-5 IDS korrekt zugeordnet wurden, einen prozentualen Anteil von 0,00008 am Gesamtvolumen aufweisen. Einträge, die von allen IDS korrekt gedeutet wurden, machen jedoch ca. 98 Prozent aus. Daraus folgt, dass die Fehlerkennungen kaum ins Gewicht fallen und die Endergebnisse der IDS sehr nah beieinander liegen. Daher wird die Aussage getroffen, dass die Bewertung nach korrekter Voraussage unpassend ist [17].

Zur Lösung dieses Problems werden neue Datensätze gebildet. Diese bestehen aus einer Auswahl von Einträgen des KDD 99 Datensatzes. Die Auswahl, welche Einträge in den Datensatz übernommen werden, hängt von den verschiedenen Schwierigkeitsgruppen ab. Der Anteil der Einträge pro Gruppe, der übernommen wird, richtet sich nach dem Anteil, den diese Gruppe am Gesamtvolumen besitzt. Macht eine Gruppe beispielsweise einen Anteil von fünf Prozent am Gesamtvolumen aus, werden 95 Prozent ihrer Einträge übernommen. Mit diesem Schritt soll die Fokussierung auf die Erkennung bestimmter Angriffe abgeschwächt werden und die Resultate sollen eine größere Varianz aufweisen. Nach diesem Schema wurden der $\text{KDDTrain}^{+}$ sowie der $\text{KDDTest}^{+}$ Datensatz erstellt. Zusätzlich zu diesen Datensätzen wurde ein Weiterer erstellt, in dem die Einträge, die von allen IDS korrekt zugeordnet wurden, entfernt wurden. Dieser trägt den Namen $\text{KDDTest}^{-21}$. Dieses Vorgehen führt zu einer noch größeren Varianz der Ergebnisse, da falsch zugeordnete Verbindungen noch stärker gewichtet werden.
Die bereitgestellten Dateien des NSL-KDD 09 Datensatzes umfassen die beiden überarbeiteten Datensätze $\text{KDDTrain}^{+}$ und $\text{KDDTest}^{+}$. Zusätzlich beinhaltet der NSL-KDD 09 einen Teildatensatz, der 20 Prozent des $\text{KDDTrain}^{+}$ umfasst und den $\text{KDDTest}^{-21}$ Datensatz.
Die bereitgestellten Datensätze liegen sowohl im *ARFF* als auch im *CSV*-Format vor. Die Zeilen entsprechen, wie auch beim KDD 99 Datensatz, jeweils einem Verbindungsvektor. Die Spaltenattribute bilden wiederum die 41 Merkmale. Die Schlüssel enthalten ein zusätzliches Attribut, das die Zuordnung zum Hintergrundverkehr oder den zugehörigen Angriff angibt.
Die Dokumentation des NSL-KDD 09 Datensatzes besteht aus einem wissenschaftlichen Artikel [17], der die Änderungen zum KDD 99 Datensatz beschreibt. Neben diesem Artikel wird keine gesonderte Dokumentation mitgeliefert.
Das Labeling umfasst die Information, ob es sich bei der jeweiligen Verbindung um Hintergrundverkehr oder einen Angriff handelt. Bei Angriffen wird der jeweilige Angriffstyp angegeben. Zusätzlich wird das Schwierigkeitsniveau, also der Wert der „successfulPrediction“ Variable angegeben.

### Kritik

*A1 Dokumentation*—Der zugehörige wissenschaftliche Artikel umfasst lediglich die Änderungen gegenüber dem KDD 99 Datensatz. Für weitere Informationen müssen die Dokumentationen der ursprünglichen Datensätze Darpa 98 und KDD 99 herangezogen werden. Somit werden auch sämtliche Schwächen dieser Dokumentationen übernommen.
Das Labeling behält ebenfalls die Schwächen der Vorgänger bei, da wiederum keine Einzelheiten zu Host und Prozess erfasst werden.

*A2 Realistische Annahmen*—Da dieser Datensatz aus Teilen des KDD 99 Datensatzes besteht, erbt er eine Vielzahl der Schwächen seiner Vorgänger. Dies sind das zu klein

modellierte Rechnernetz und die geringe Anzahl von Hosts, mit denen interaktive Angriffe durchgeführt werden.
Ebenfalls kommen nur Angriffe von Außerhalb des überwachten Netzes vor.
Die Auswahl der Merkmale für die Verbindungsvektoren wird beibehalten und somit bleibt auch die in 2.3.2 genannte Kritik an deren Relevanz für die Erkennung von Angriffen bestehen.
Verbesserungen wurden im Punkt der redundanten Einträge erzielt, da diese komplett aus dem Datensatz entfernt wurden. Somit wurde eine ausgewogene Gewichtung der unterschiedlichen Verbindungen erreicht.
Des Weiteren wurde eine Verbesserung bei der hohen Rate der korrekten Zuordnungen durch die IDS erzielt. Durch den höheren Anteil von Verbindungen, die schwerer zuzuordnen sind, ergeben sich größere Unterschiede bei den Evaluationsergebnissen der einzelnen IDS. Infolge dessen fällt es leichter, Aussagen über die Qualität eines IDS gegenüber anderen zu treffen.

*A3 Interaktion zwischen Angriffen und Hintergrundverkehr*—Da die Daten nicht neu erstellt werden, sondern aus denen der ursprünglich beim Darpa 98 erstellten Daten bestehen, findet eine Interaktion auch bei diesem Datensatz nicht statt. Die Kritik aus 2.3.1 und 2.3.2 zu diesem Punkt bleibt bestehen.

## 2.4 Zusammenfassung

In Bezug zwischen Anforderungen und der Kritik an den Datensätzen lässt sich zusammenfassen, dass die vorhandenen Datensätze in vielerlei Hinsicht nützlich sind und dies auch unter Beweis gestellt haben. Problematisch ist jedoch, dass die genannten Datensätze alle auf einem ursprünglichen Datensatz basieren, der allein schon eine Vielzahl von Kritikpunkten auf sich vereint.
Beginnend bei der Dokumentation des Darpa 98 Datensatzes, die viele für die Replizierung relevanten Einzelheiten vermissen lässt, gelingt es auch den folgenden Datensätzen nicht, dies genauer zu beschreiben. Vielmehr verweisen diese Datensätze lediglich auf die Dokumentation des Darpa 98 Datensatzes, indem die Beschreibung der Umgebung und der Datenaufzeichnung nur oberflächlich erfolgt oder komplett fehlt. Es werden lediglich die Änderungen zum vorhergehenden Datensatz thematisiert.
Die Schwächen beim Labeling werden ebenfalls nicht ausgebessert. Alle genannten Datensätze bestimmen lediglich, ob das Paket bösartig ist und zu welchem Angriffstyp es gehört.
Da nach dem Darpa 98 Datensatz keine neuen Daten erzeugt wurden, basieren alle Datensätze auf der Annahme, dass Angriffe nur von außerhalb des Netzes stattfinden. Dies ist in Bezug auf die aktuellen Angriffe in 2.1 inakzeptabel, denn unter aktuellen Annahmen ist ein Angriff von innerhalb des Netzes ein durchaus wahrscheinliches Szenario. Somit ist ein Großteil der aktuell relevanten Angriffe in den vorhandenen Datensätzen

nicht enthalten. Gleiches gilt für die mangelnde Interaktion von Hintergrundverkehr und Angriffen, die ebenfalls bei allen Datensätzen präsent ist.
Für die weiteren Kritikpunkte zu den realistischen Annahmen gilt, dass lediglich im NSL-KDD09 Datensatz versucht wird, den Datensatz realistischer zu gestalten. Dies gelingt jedoch auch nur teilweise, da beispielsweise die Größe des Netzes oder die Anzahl von in Angriffen involvierte Rechner nicht im Nachhinein verändert werden können.
Dieses Fazit, dass die vorhandenen Datensätze einerseits nützlich bei der Entwicklung von IDS sind, andererseits jedoch schon bei Erstellung Schwächen aufwiesen und auf einem 16 Jahre altem Datensatz basieren, der mittlerweile unter mehreren Gesichtspunkten als veraltet angesehen werden kann, führen zu der Forderung nach der Erstellung neuer Datensätze.
In Tabelle 2.1 sind die Datensätze und die jeweils erfüllten Anforderungen eingetragen. Das „X“ bedeutet, dass zu dieser Anforderung gravierende Schwächen im Datensatz auftreten. Das „-“ besagt, dass die Anforderungen nur teilweise umgesetzt sind.

Tabelle 2.1: Erfüllte Anforderungen

| | Darpa 98 | KDD 99 | NSL-KDD 09 |
|---|---|---|---|
| A1 | - | X | X |
| A2 | X | X | - |
| A3 | X | X | X |

# Kapitel 3

# Entwurf

Im folgenden Kapitel soll nun die Erstellung neuer Datensätze im Vordergrund stehen. Es wird anhand der Anforderungen aus Kapitel 2.2 definiert, welche Kriterien bei der Erstellung erfüllt werden müssen. Die Erstellung soll mithilfe eines Frameworks umgesetzt werden. Im Folgenden werden zunächst die Funktionen, die dieses Framework bereitstellen muss, beschrieben. Daraufhin wird der Aufbau des Frameworks beschrieben. In den weiteren Abschnitten wird dann auf die einzelnen Teilbereiche des Frameworks näher eingegangen.

## 3.1 Framework

Zunächst werden die Anforderungen aus den Bereichen Dokumentation, realistische Annahmen und Interaktion zwischen Angriffen und Hintergrundverkehr betrachtet um zu definieren, welche Funktionalitäten zu deren Erfüllung notwendig sind. Darauffolgend wird der Aufbau, der die Umsetzung der Funktionalitäten sowie die Erzeugung individueller Datensätze erlauben soll, vorgestellt

### 3.1.1 Funktionalitäten

Das Framework zur Erstellung neuer Datensätze muss so konzipiert werden, dass bei Erzeugung der Datensätze die in 2.2 formulierten Anforderungen berücksichtigt werden. Um die detaillierte Dokumentation sicher zu stellen, werden Funktionen gefordert, die sämtliche Parameter der verwendeten Software sowie der Testumgebung erfassen. Darüber hinaus muss der Netzverkehr mitgeschnitten werden. Dieses Logging muss so realisiert werden, dass eine spätere Zuordnung einzelner Pakete zum Hintergrundverkehr sowie den Angriffen möglich ist. Am Ende der Aufzeichnung muss die Zuordnung der Pakete durchgeführt werden, um einen Schlüssel für die Testdaten zu erzeugen.
Um realistische Szenarien erzeugen zu können, sind Funktionen, die sowohl Angriffe als auch Hintergrundverkehr erzeugen, notwendig. Um eine Interaktion von Angriffen und Hintergrundverkehr zu erhalten, müssen diese zur Laufzeit der Aufzeichnung real erzeugt werden.

Um den Anforderungen an den Angriffsverkehr gerecht zu werden, muss der Teil des Frameworks, der für dessen Erzeugung vorgesehen ist, zwei Kriterien erfüllen. Ein Kriterium ist, dass eine Mehrzahl von Angriffen aus verschiedenen Klassen zur Ausführung vorhanden ist. Das andere Kriterium ist das Vorkommen aktuell relevanter Angriffe. Für die Auswahl dieser Angriffe gelten die in 2.2 ausgeführten Anforderungen. Die Kritik an vorhandenen Datensätzen hat gezeigt, dass selbst wenn dieser Punkt bei Erstellung der Datensätze berücksichtigt und angemessen umgesetzt wurde, mit zunehmendem Alter des Datensatzes diese aktuelle Relevanz schwindet. So muss also eine Erweiterbarkeit bei der Auswahl der Angriffe möglich sein, um die Erzeugung von Datensätzen mit derzeit aktuell relevanten sowie zukünftig aktuell relevanten Angriffen zu ermöglichen.
Der Teil des Frameworks, der die Erzeugung des Hintergrundverkehrs steuert, muss ebenfalls zwei Kriterien erfüllen. Es muss bei Erstellung des Hintergrundverkehrs eine Auswahl aus mehreren Protokollen zur Verfügung stehen, damit Angriffe realistischer eingebettet werden können. Das weitere Kriterium ist, dass der Hintergrundverkehr nicht zu simpel sein darf. So darf sich beispielsweise die Implementierung von *HTTP*-Verkehr nicht auf das mehrfache Aufrufen ein und der selben Webseite beschränken.
Die Anforderungen, die Topologie und Datenrate betreffen, erfordern, dass das Framework auch in Umgebungen mit vielen Rechnern einsetzbar ist und das die einzelnen Funktionen, wie beispielsweise das Logging, auch bei hohen Datenraten funktionieren.

### 3.1.2 Aufbau

Der Aufbau des Frameworks soll die Erstellung einer Vielzahl von Testdatensätzen ermöglichen. So soll dem Anwender die Möglichkeit gegeben werden, individuelle Szenarien zu erstellen. Um dies zu gewährleisten, soll vor Erzeugung der Datensätze die Auswahl der anzuwendenden Angriffe möglich sein. Um das jeweilige Zielnetz besser modellieren zu können, soll auch der Hintergrundverkehr anpassbar sein. So sollen Menge und Typ des Hintergrundverkehrs ausgewählt werden können. Dies führt dazu, dass der Anwender selbst entscheiden kann, auf welche Angriffe er den Schwerpunkt legt und in welchen Hintergrundverkehr diese eingebettet werden. Zusätzlich können Datensätze mit ähnlichen Szenarien erzeugt werden. Dies ermöglicht es, die Lernalgorithmen von anomaliebasierten IDS effektiver zu testen, da geprüft werden kann, ob auch ähnliche Versionen zu bereits bekannten Angriffen erkannt werden.
Ein weiterer Vorteil dieses flexiblen Konzepts des Frameworks ist die Möglichkeit, Erweiterungen leichter einbinden zu können. Das ermöglicht die Aufnahme neuer Angriffe in die Auswahl ohne großen Aufwand und infolgedessen die Erstellung neuer Datensätze, die aktuell relevante Angriffe beinhalten. Diese Erweiterbarkeit gilt ebenfalls für den Hintergrundverkehr, so dass auch dort aktualisierte oder neue Dienste eingebunden werden können.
Das Framework soll zusätzlich so implementiert werden, dass es unabhängig von der Hardware und Topologie der Testumgebung ist. So soll bei Ausführung des Frameworks keine besondere Hardware erforderlich sein. Dies erlaubt, dass die Erstellung der Daten-

sätze nicht an eine bestimmte Umgebung gebunden ist, sondern überall erfolgen kann. Zusätzlich können durch die Unabhängigkeit von einer bestimmten Topologie noch individuellere Szenarien erstellt werden, da neben Angriffen und Hintergrundverkehr auch die Rechner in der Testumgebung wählbar sind.
Zur Umsetzung der einzelnen Aspekte wird ein modularer Aufbau des Frameworks gewählt (Abb. 3.1). Dieser besteht aus drei Komponenten: Einem Angriffsmodul, das die Ausführung der Angriffe steuert, einem Hintergrundverkehrsmodul, das die Erzeugung des Hintergrundverkehrs verwaltet, sowie einem Logging-Modul. Dieses Modul ist für die Datenaufzeichnung und Datensammlung während des Experiments zuständig. Im Folgenden werden die Begriffe Komponente und Modul synonym verwendet.

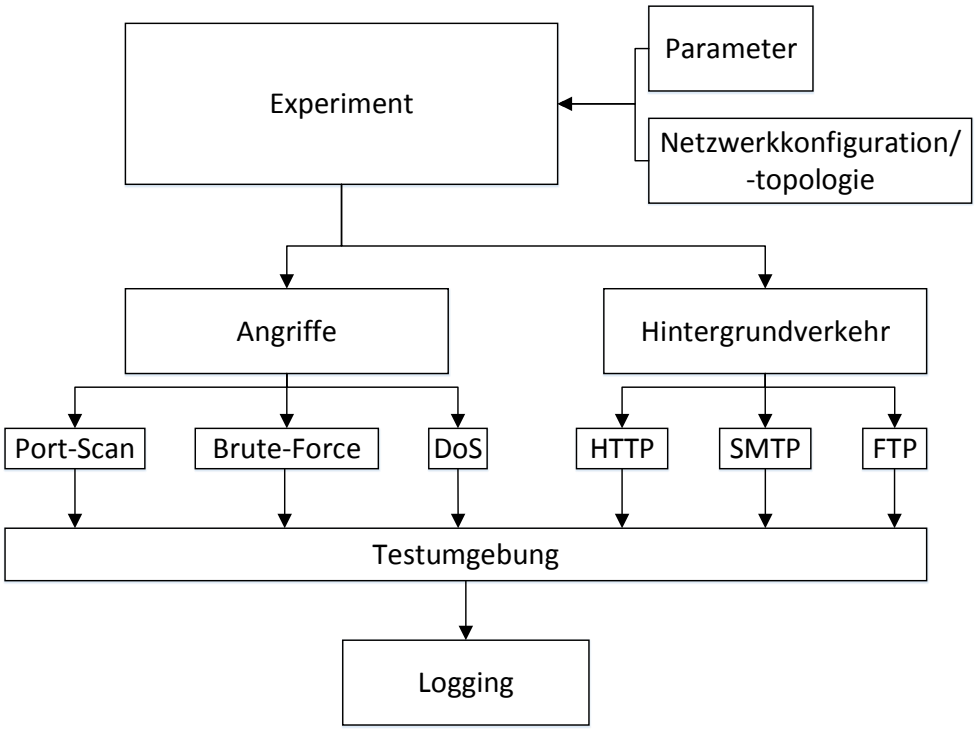


Abbildung 3.1: Modulares Framework

Zu Beginn der jeweiligen Aufzeichnung, dem Experiment, definiert der Benutzer die zu übergebenden Parameter. Dies sind unter anderem die Dauer der Aufzeichnung, die Auswahl der Angriffe sowie die Auswahl des Hintergrundverkehrs. Darüber hinaus muss während des Experiments die Netzwerktopologie bekannt sein, um Informationen über die im Netz befindlichen Rechner zu erhalten.
Sind diese Eingaben erfolgt, soll die Durchführung des Experiments mit Hintergrundverkehr und Angriffen sowie das Aufzeichnen der relevanten Daten automatisch erfolgen. Zur Laufzeit des Experiments werden die zuvor spezifizierten Parameter an die jeweiligen Module weitergegeben. Diese starten daraufhin die für das Experiment gewünschten

Dienste und führen ihre Aufgaben in der Testumgebung aus. Im Beispiel (Abb. 3.1) werden drei Angriffe durchgeführt (Port-Scan, Brute-Force, DoS) und drei verschiedene Protokolle für die Erzeugung von Hintergrundverkehr verwendet (*HTTP*, *SMTP*, *FTP*). Die dadurch erzeugten Daten werden durch das Logging-Modul aus der Testumgebung aufgezeichnet. Der Informationsfluss, beginnend mit den Eingaben des Benutzers bis zur Erzeugung der Daten in der Testumgebung, wird durch Pfeile dargestellt. Am Ende des Experiments soll dann der Datensatz im *.pcap*-Format sowie dessen Schlüssel vorliegen. Zusätzlich sollen sämtliche Informationen, die für die Dokumentation benötigt werden, aufgezeichnet sein.

## 3.2 Komponenten

Durch die Strukturierung mit mehreren Komponenten soll das Framework leichter zu modifizieren sein. Die einzelnen Module sollen unabhängig voneinander sein, so dass bei Bedarf ganze Komponenten überarbeitet oder ausgetauscht werden können. Auf die genannten Module wird im Folgenden näher eingegangen.

### 3.2.1 Angriffe

Das Angriffsmodul soll die Durchführung der einzelnen Angriffe realisieren. Alle Aspekte der Angriffe sollen in diesem Modul vereint werden. Dem Anwender soll die Möglichkeit gegeben werden, aus einer Auswahl von Angriffen die Gewünschten zu selektieren und in der Testumgebung auszuführen. Für jeden Angriff spaltet sich die Durchführung in drei Schritte.
Im Folgenden wird beispielhaft die Umsetzung eines Brute-Force-Angriffs betrachtet. Bei diesem wird versucht, Zugriff auf ein System zu erhalten, dass eine Authentifikation mittels Benutzernamen und Passwörtern durchführt. Im Rahmen der Brute-Force-Attacke werden alle möglichen Kombinationen von zuvor festgelegten Benutzernamen und Passwörtern getestet.
Ziel bei der Erstellung der Komponente ist, dass der Benutzer vor Beginn des Experiments neben der Auswahl der Angriffe auch deren Parameter festlegt und der weitere Ablauf automatisch erfolgt. Zu den Parametern, die zu spezifizieren sind, zählt der Ausgangspunkt des Angriffs. Dieser legt fest, auf welchem Rechner die Software der Angriffskomponente ausgeführt wird. Zusätzlich muss ein Ziel festgelegt werden, das der Angriffsrechner attackieren soll. Bei einem Brute-Force-Angriff muss zudem noch das Protokoll angegeben werden, das auf dem Zielrechner angegriffen werden soll. Während des Experiments können Angriffe auch mehrfach ausgeführt werden. Aus diesem Grund muss der Benutzer zu Beginn die Zeitpunkte der Wiederholungen angeben.
Um diese Ziele zu erreichen, müssen im ersten Schritt zunächst die Vorbereitungen getroffen werden. Hierzu zählt die Installation von benötigter Software. Dies umfasst die Software, die für den Angriff selbst benötigt wird, im Beispiel das Brute-Force-Programm, aber auch die zu attackieren Software auf dem Zielrechner. Darüber hinaus

müssen weitere für den Angriff benötigte Dateien platziert werden. Bei einem Brute-Force-Angriff sind dies Listen mit den zu testenden Passwörtern und Benutzernamen. Die Vorbereitungen müssen vor der Durchführung des Experiments erfolgen.
Der zweite Schritt wird während der Messungen ausgeführt. Er umfasst den eigentlichen Angriff. Dieser kann mehrfach durchgeführt werden. Da der Benutzer bei einem Brute-Force-Angriff lediglich die allgemeinen Parameter *Angreifer*, *Ziel* und *Protokoll* festlegen soll, müssen detailliertere Parameter durch das Angriffsmodul festgelegt werden. Die zuvor platzierten Listen mit den Passwörtern und Benutzernamen müssen also automatisch eingebunden werden. Darüber hinaus müssen der Aufruf des Programms mit allen Parametern aufgezeichnet und mögliche Ausgaben protokolliert werden.
In der Nachbearbeitung, dem dritten Schritt, müssen die Programme gestoppt werden, falls diese nicht bereits während des Experiments beendet wurden. Im Anschluss beginnt die Aufräumphase. Die zuvor platzierten Dateien müssen wieder gelöscht und erzeugte Protokolldateien eingesammelt und dem Logging zugeführt werden. Am Ende dieser Phase sollen die in den Angriff involvierten Rechner wieder in den ursprünglichen Zustand zurückversetzt werden, in dem sie sich vor dem Experiment befanden. Sollte die für den Angriff verwendete Software nicht dauerhaft für den Verbleib auf den Testrechnern vorgesehen sein, muss auch diese wieder deinstalliert werden.

### 3.2.2 Hintergrundverkehr

Dieses Modul beinhaltet die Generierung des Hintergrundverkehrs. Es soll die Auswahl zwischen verschiedenen Diensten bieten. Im Diagramm (Abb. 3.1) sind stellvertretend *HTTP*, *SMTP* und *FTP* aufgeführt. Ziel ist es jedoch, auch weitere Dienste bereitzustellen und Erweiterungen des Moduls zu ermöglichen.
Im Gegensatz zum Darpa 98 Datensatz soll der Hintergrundverkehr real erzeugt werden. Um dies zu erreichen, wird der Verkehr von den vorhandenen Hosts im Netzwerk erzeugt. Dafür müssen auf diesen die notwendigen Dienste ausgeführt werden. Die direkte Erzeugung ermöglicht es, die Anforderung nach Interaktion von Angriffen und Hintergrundverkehr zu erfüllen. Ein Host, der realen Verkehr erzeugt und angegriffen wird, reagiert gezwungenermaßen auf diesen Angriff. Einerseits können so DoS-Angriffe realistisch dargestellt werden und andererseits ist jeder Host im Netzwerk als potentielles Ziel für einen interaktiven Angriff wählbar.
Neben dem realistischeren Verhalten bei Angriffen werden auch Aspekte für das realistische Verhalten des Hintergrundverkehrs bei der direkten Erzeugung automatisch berücksichtigt. So können Effekte wie die „Internet background noise“ (IBN) auftreten, ohne dass diese künstlich modelliert werden müssen.
Wie schon beispielhaft bei den Angriffen veranschaulicht wurde, findet auch bei der Erzeugung des Hintergrundverkehrs eine Aufteilung in drei Schritte statt. Diese sind ebenfalls Vorbereitung, Durchführung und Nachbearbeitung. Bei der Spezifizierung der Parameter legt der Benutzer *Start* und *Ziel* der Verbindungen sowie *Typ* des Hintergrundverkehrs fest. Die weiteren Parameter, wie beispielsweise *Ports* und *Zugangsdaten*

für die Verbindung zum Zielrechner, werden durch das Modul verwaltet.
Während der Vorbereitung werden wiederum die benötigten Programme eingerichtet und Dateien für die spätere Nutzung (z.B. in *FTP*-Downloads) platziert. Nach der Erzeugung des Hintergrundverkehrs während der Durchführung, werden in der Nachbearbeitung sämtliche Prozesse gestoppt und alle zuvor erfolgten Änderungen an den Rechnern werden zurückgesetzt.

### 3.2.3 Logging

Das Logging-Modul muss den gesamten Bereich der Datenaufzeichnung umfassen. Zu diesen Daten zählt die Aufzeichnung des Netzwerkverkehrs. Dieser muss an bestimmten Messpunkten (Abb. 3.2) erfasst werden und bildet die Grundlage des Datensatzes. Es müssen externe Angriffe beispielsweise aus dem Internet aufgezeichnet werden können. Zusätzlich müssen auch Szenarien mit Angreifern aus dem Intranet, wie in 2.1 beschrieben, modelliert werden können. Daher müssen sich die Messpunkte sowohl außerhalb als auch innerhalb des zu schützenden Netzes befinden.
Neben den Netzverkehrsdaten müssen auch alle weiteren, für die Dokumentation relevanten Einzelheiten, aufgezeichnet werden. Hierzu zählen die während der Aufzeichnung durchgeführten Angriffe, um später die bösartigen Pakete korrekt beschriften zu können. Zusätzlich muss ein Netzplan erstellt oder als Parameter übergeben werden, um die Topologie der Testumgebung später nachvollziehen zu können. Um eine Replizierung des Experiments zu erlauben, müssen sämtliche Parameter aufgezeichnet werden. Das sind einerseits die Parameter, die der Benutzer eingibt und andererseits die Parameter, mit denen letztendlich die Programme durch das Framework aufgerufen werden.
Im Vorbereitungsteil müssen Ordner angelegt werden, in denen später die Aufzeichnungen gespeichert werden. Zusätzlich müssen alle weiteren Einstellungen vorgenommen werden, die für für das Logging nötig sind. Dies beinhaltet die Installation von Software sowie deren Konfiguration.
Zu Beginn der Durchführung folgt das Starten der Aufzeichnung.
Am Ende der Aufzeichnung müssen alle Daten in die vorgesehenen Ordner zusammengetragen werden. Gemäß den Anforderungen muss im Rahmen der Nachbearbeitung eine Zuordnung der einzelnen Pakete zu Quell-Host und zugehörigem Prozess stattfinden. Während der Aufräumarbeiten müssen auch alle zuvor für das Logging geänderten Einstellungen rückgängig gemacht werden.

### 3.2.4 Ausführung

Damit der Benutzer nur zu Beginn Eingaben tätigen muss und die weitere Durchführung des Experiments automatisch abläuft, muss eine Komponente den Ablauf des Experiments steuern. Somit muss neben den Komponenten für Angriffe, Hintergrundverkehr und Logging ein zusätzliches Steuerungsmodul vorhanden sein. Diese Komponente kon-

trolliert den Ablauf des Experiments und ruft die einzelnen Schritte der Komponenten zum richtigen Zeitpunkt auf (Abb. 3.2).

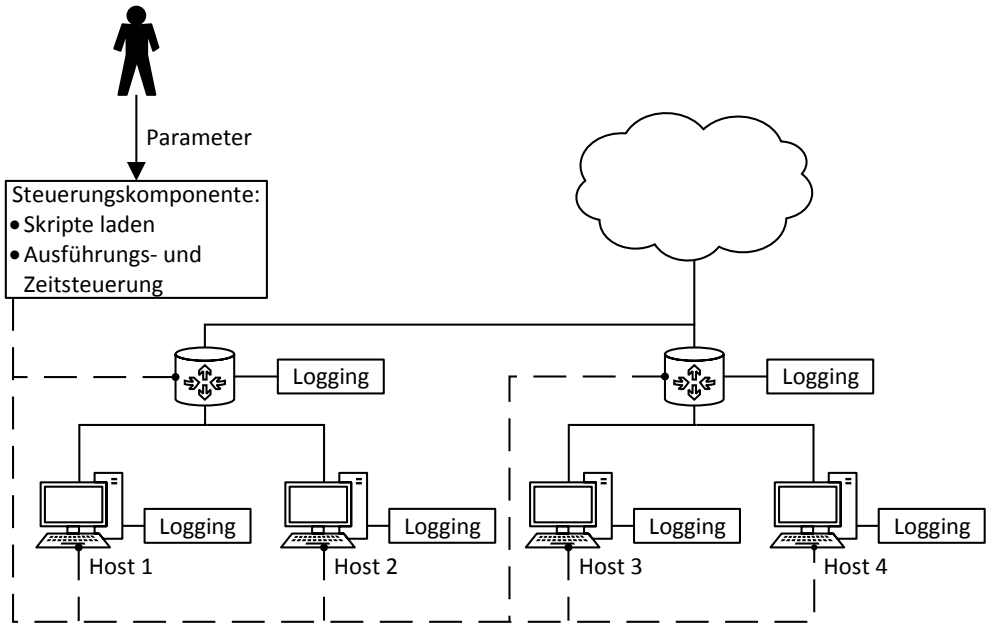


Abbildung 3.2: Testumgebung

Der modulare Aufbau des Frameworks wird durch die Verwaltung der Programmteile aller Komponenten in separaten Dateien zusätzlich unterstützt. Diese Vorgehensweise erlaubt die gewünschte Unabhängigkeit zwischen den Komponenten. Die einzelnen Module können ohne Beeinflussung der anderen modifiziert werden. Um die Flexibilität auch innerhalb der Komponenten zu gewährleisten, wird zusätzlich jeder der drei Schritte in einzelnen Skripten verwaltet. So besteht jeder Angriff sowie jeder Typ von Hintergrundverkehr aus drei separaten Programmteilen.
Während der Vorbereitung müssen durch die Steuerungskomponente zuerst die einzelnen Skripte der anderen Module auf den Rechnern platziert werden, die für deren Ausführung vorgesehen sind. Anhand der zuvor eingegebenen Parameter wird festgelegt, welche Skripte auf welchem Host benötigt werden. Die verteilte Ausführung der Skripte auf den verschiedenen Rechnern wird durch die Steuerungskomponente koordiniert.
Zuerst müssen die Teile der Komponenten gestartet werden, die zur Vorbereitung auf das Experiment dienen. Am Ende der Vorbereitung müssen alle für das Experiment benötigten Dienste gestartet sein, wie beispielsweise Webserver für die Erzeugung von *HTTP*-Verkehr. Zusätzlich müssen alle benötigten Dateien, wie Listen für Brute-Force-Angriffe, infizierte Dateien für den Download von Malware und die Inhalte auf *FTP*-Servern platziert worden sein

Sind die Vorbereitungen erfolgt, so muss zu Beginn der Durchführung die Erzeugung des Hintergrundverkehrs sowie die Aufzeichnung des Logging-Moduls gestartet werden. Die Ausführung der Angriffe wird ebenfalls gesteuert. Jedoch werden diese in der Regel nicht alle sofort zu Beginn des Experiments durchgeführt. Somit muss die Steuerungskomponente den verzögerten Start der Angriffe zu einem zuvor festgelegten Zeitpunkt sowie die mögliche mehrfache Ausführung regeln.
Die Steuerungskomponente muss auch das Ende des Experiments einleiten. Dazu müssen die Teile der drei weiteren Module aufgerufen werden, die für die Nachbereitung zuständig sind.
Dieser Ablauf mit der Aufteilung in drei Teilschritte ist in Abbildung 3.3 schematisch dargestellt. Das Starten und Stoppen des Loggings ist separat aufgeführt, da Beginn und Ende der Durchführungsphase von diesen beiden Zeitpunkten definiert werden. Der Verweis der Durchführung auf sich selbst stellt das mehrmalige Ausführen von Programmteilen während dieses Schrittes dar.

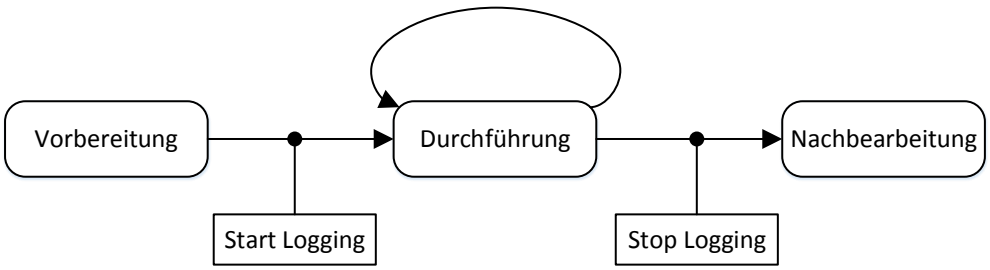


Abbildung 3.3: Experiment Ablauf

# Kapitel 4

# Implementierung

In diesem Kapitel wird zunächst das Vorgehen während der Implementierungsphase dargelegt. Es wird kurz begründet, warum zuerst die Logging-Komponente implementiert wurde. Daraufhin wird eine Auswahl möglicher Ansätze zur Umsetzung der für das Framework geforderten Komponenten vorgestellt und analysiert. Es wird dann die Umsetzung des gewählten Ansatzes im Detail vorgestellt. Abschließend wird auf die Eclectic Umgebung eingegangen, die zur Umsetzung der Steuerungskomponente geplant ist.

## 4.1 Logging

Zu Beginn der Implementierung wurden zunächst die zu implementierenden Arbeitsschritte analysiert. Bei dieser Analyse wurde eine Gewichtung nach der Priorität durchgeführt. Als Ergebnis dieser Analyse wurde das Logging als vorrangig zu implementierender Arbeitsschritt festgelegt. Die Begründung für diese Wahl liegt darin, dass das Logging, speziell das Labeling der einzelnen Pakete, ein elementarer Bestandteil bei der automatischen Erzeugung von Testdatensätzen ist. Wird das Labeling nicht sorgfältig durchgeführt, sind funktionierende Angriffs- oder Hintergrundverkehrsmodule zweitrangig, da selbst bei Datensätzen, die alle Anforderungen in den Bereichen Angriffe und Hintergrundverkehr erfüllen, nicht verifiziert werden kann, ob das IDS Pakete korrekt erkannt hat.
Das erste Ziel der Implementierungen war die Zuordnung von Netzwerkpaketen zu dem erzeugenden Prozess auf dem Quellrechner. Wenn der erzeugende Prozess bekannt ist, sind auch die weiteren für das Labeling erforderlichen Informationen bekannt. Diese beinhalten den Host, auf dem das Paket erzeugt wurde. Bei Angabe des erzeugenden Prozesses muss der Host zwangsläufig bekannt sein. Des Weiteren kann bei Angabe des Prozesses auch dessen Aufgabe angegeben werden. Dies soll sowohl für Angriffe als auch für Hintergrundverkehr erfolgen. Mithilfe dieser Zuordnung kann das Labeling so detailliert, wie in 2.2 gefordert, erfolgen.
Früh stellte sich diese Aufgabe als komplizierter heraus, als von vornherein angenom-

men. Bereits vorhandene Werkzeuge, die diese Aufgabe übernehmen, konnten nicht gefunden werden. Es wurden Programme gefunden, welche die aktuellen Sende- und Empfangsraten pro Prozess angeben, jedoch nicht die einzelnen Pakete.
Somit wurden Programme gesucht, aus deren Informationen die Verbindung zwischen Paket und Prozess rekonstruiert werden kann.

### 4.1.1 Netstat/SS

Erstes Ergebnis dieser Suche war das Kommandozeilenprogramm *Netstat* sowie dessen Nachfolger *SS* (Socket Stat). Diese Werkzeuge erlauben es, die aktuell offenen Verbindungen des Rechners darzustellen (Abb. 4.1). Die für eine Zuordnung relevanten Informationen, die *Netstat/SS* liefern, sind Prozessname und PID sowie der zugehörige Port. Mithilfe dieser Informationen ist eine Zuordnung von Ports und Prozessen möglich. Pakete können so über den Schlüssel Port mit den zugehörigen Prozessen verknüpft werden. Dies gelingt durch Auslesen der Ports aus den einzelnen Paketen und der Suche der zugehörigen Portnummer in den *Netstat/SS* Aufzeichnungen.
Nachteil dieser Variante ist, dass die Zuordnung lediglich über die Portnummer erfolgt. Somit wäre ein Labeln nur von *TCP* und *UDP* Paketen möglich. Es müsste also ohnehin eine weitere Methode gefunden werden, Pakete unterhalb der Transportschicht zu markieren. Hinzu kommt, dass der Aufruf von *Netstat* oder *SS* zeitgesteuert erfolgt. Dies bedeutet, dass alle Änderungen, die zwischen den jeweiligen Aufrufen stattfinden und wieder beendet sind, nicht aufgezeichnet werden. Es kann also passieren, dass Pakete nicht zuzuordnen sind, da kein Eintrag in den *Netstat/SS* Aufzeichnungen vorhanden ist. Darüber hinaus können Ports in bestimmten Zeitintervallen wiederholt von unterschiedlichen Prozessen genutzt werden. Infolgedessen reicht der Port allein als Schlüssel nicht aus. Auch der Zeitstempel muss berücksichtigt werden. Aus diesem Grund wurde im weiteren Verlauf ein Ansatz gesucht, der ereignisgesteuert arbeitet und somit alle Änderungen erfasst werden.

| Proto | Recv-Q | Send-Q | Local Address | Foreign Address | State | PID/Program name |
|---|---|---|---|---|---|---|
| tcp | 0 | 0 | 127.0.1.1:53 | 0.0.0.0:* | LISTEN | 996/dnsmasq |
| tcp | 0 | 0 | 127.0.0.1:631 | 0.0.0.0:* | LISTEN | 2157/cupsd |
| tcp | 0 | 0 | 10.0.2.15:35316 | 192.168.178.20:139 | ESTABLISHED | 5958/gvfsd-smb-brow |
| tcp | 0 | 0 | 10.0.2.15:46795 | 192.168.178.21:139 | ESTABLISHED | 5958/gvfsd-smb-brow |
| tcp | 0 | 0 | 10.0.2.15:59179 | 74.125.232.90:443 | ESTABLISHED | 4164/firefox |
| tcp | 0 | 0 | 10.0.2.15:46850 | 173.194.44.82:443 | ESTABLISHED | 4164/firefox |
| tcp | 0 | 0 | 10.0.2.15:46098 | 173.194.44.6:443 | ESTABLISHED | 4164/firefox |
| tcp | 0 | 0 | 10.0.2.15:35319 | 192.168.178.20:139 | ESTABLISHED | 5958/gvfsd-smb-brow |
| tcp | 0 | 0 | 10.0.2.15:49049 | 74.125.232.89:443 | ESTABLISHED | 4164/firefox |
| tcp | 0 | 0 | 10.0.2.15:56613 | 74.125.232.73:443 | ESTABLISHED | 4164/firefox |

Abbildung 4.1: Ausgabe von *Netstat*

Der nächste Ansatz bestand darin, zu ermitteln, woher *Netstst/SS* ihre Informationen beziehen. So wurde versucht, mit dem Kommandozeilenprogramm *strace* herauszufin-

den, auf welche Dateien *Netstat/SS* bei ihrem Aufruf zugreifen. Es konnte herausgestellt werden, dass auf eine Vielzahl von Dateien in den Systemverzeichnissen, unter anderem */proc*, zugegriffen wird. Durch die Vielzahl einzelner Aufrufe von Dateien und Funktionen konnten die Daten nicht ohne weiteres rekonstruiert werden. Somit wurde dieser Ansatz nicht weiter verfolgt.

### 4.1.2 Control Groups

Ein weiterer vielversprechender Ansatz ist die Nutzung von „control groups“ (*cgroups*) in Verbindung mit der *iptables*-Firewall. Seit der Linux Kernel Version 2.6.24 [23] erlauben es *cgroups*, Prozesse zu gruppieren und den Ressourcenzugriff dieser Gruppen zu verwalten. Alle Kindprozesse der zugeordneten Prozesse werden ebenfalls der *cgroup* der übergeordneten Prozesse zugeordnet. Die Funktionalitäten der *cgroups* beinhalten die Beschränkung des Ressourcenzugriffs, die Vergabe von Prioritäten sowie die Isolation von Prozessgruppen. Darüber hinaus können auch Netzwerkpakete einzelnen *cgroups* zugeordnet werden.
Für die Zuordnung eines Netzwerkpakets zu einer *cgroup* wird *iptables* benötigt. *Iptables* ermöglicht die Erstellung von Regeln für das Filtern von Netzwerkpaketen. Dies wird häufig im Rahmen einer Firewall eingesetzt, um unerwünschten Netzwerkverkehr zu blockieren. Der Aufbau besteht aus mehreren Ketten (Abb. 4.2), für die Filterregeln definiert werden können [24]. Wichtig im Rahmen dieses Ansatzes ist die *INPUT*-Kette. Die Regeln dieser Kette werden auf alle eingehenden Pakete angewendet, die an einen Prozess auf dem Rechner gerichtet sind. Darüber hinaus wird die *OUTPUT*-Kette benötigt, deren Regeln auf alle ausgehenden Pakete angewendet werden, die von einem Prozess auf dem Rechner versandt wurden.

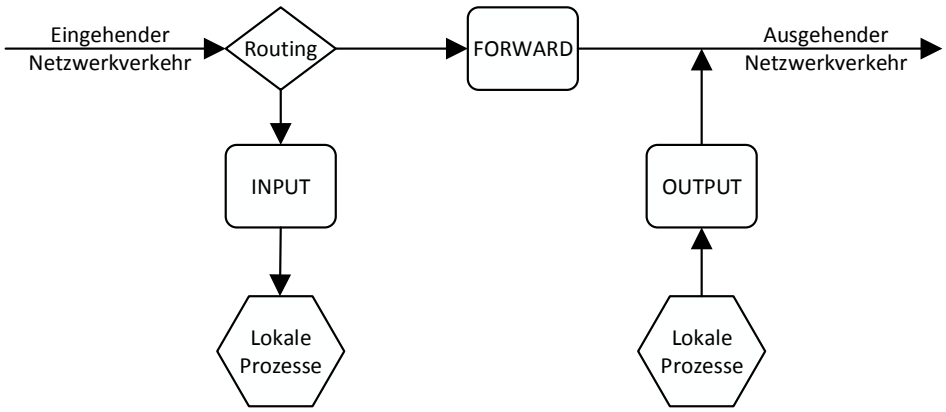


Abbildung 4.2: Aufbau *iptables* nach [24]

Um Pakete einer *cgroup* zuzuordnen muss eine Regel der *OUTPUT*-Kette hinzugefügt werden, die mittels der Option *-m cgroup* prüft, ob die ausgehenden Pakete zu der

angegebenen *cgroup* gehören. Ist dies der Fall, werden mithilfe von „connection tracking“ sämtliche Pakete dieser Verbindung markiert. Dies beinhaltet sowohl ausgehende als auch eingehende Pakete der Verbindung. Die markierten Pakete werden dann mittels einer Regel in der *OUTPUT*- sowie in der *INPUT*-Kette anhand ihrer Markierung in eine speziell für die *cgroup* angelegte *nflog*-Gruppe weitergeleitet. Dies kann für mehrere *cgroups* gleichzeitig erfolgen. Im folgenden Schritt werden alle markierten Pakete in den verschiedenen *nflog*-Gruppen mithilfe eines Programms wie *Tcpdump* oder *Dumpcap* aufgezeichnet. Die Pakete jeder *nflog*-Gruppe werden in separate Dateien im *.pcap*-Format gespeichert.

Um unter Verwendung dieses Ansatzes Pakete einem Prozess zuzuordnen, werden zunächst verschiedene *cgroups* angelegt. Den Gruppen werden anschließend Programme zugeordnet. Damit Pakete als Teil eines Angriffs markiert werden können muss mindestens eine Gruppe angelegt werden, in der die Angriffsprozesse ausgeführt werden. Bei mehreren Angriffen in einer *cgroup* lassen sich Pakete den einzelnen Angriffen möglicherweise nicht eindeutig zuordnen. Aus diesem Grund ist für die detaillierte Zuordnung eine *cgroup* pro Angriff optimal. Soll der Hintergrundverkehr auch differenziert gelabelt werden, müssen auch dessen Prozesse in verschiedene *cgroups* eingeteilt werden. Durch die Nutzung ausreichend vieler *cgroups* ist es demnach möglich die Pakete so genau zuzuordnen, dass das in 2.2 spezifizierte Detaillevel erreicht werden kann.

Nachteil dieser Lösung und Kriterium für den Ausschluss ist ihre Software-Anforderung. Um die Funktion von *iptables* nutzen zu können, die prüft, ob Pakete zu einer *cgroup* gehören, wird die neueste Version von *iptables* (Ver. 1.4.21) mit dem Modul *xt_cgroup* benötigt [25]. Diese benötigt ihrerseits die *libnftnl*-Bibliothek, die mindestens die Linux Kernelversion 3.14 [26] erfordert. Dieser kommt allerdings erst im Kernel der Ubuntu Version 14.10 zum Einsatz. Bei dieser Version handelt es sich jedoch nicht um eine Version mit Langzeitunterstützung (LTS). In der Testumgebung wird indes aus Stabilitätsgründen ausschließlich eine LTS Version (Ver. 14.04 LTS) benutzt. Somit ist die Funktion auf *cgroups* zu prüfen derzeit noch nicht nutzbar.

### 4.1.3 PID matching

Neben dem Prüfen auf *cgroups* lässt *iptables* auch weitere Parameter zu, auf die geprüft werden kann. So bot *iptables* in früheren Versionen das Prüfen auf Prozess IDs (PID). Dies würde die Problemstellung angemessen lösen, da eine Verbindung von Netzwerkpaketen und dem erzeugenden Prozessen direkt hergestellt werden könnte.

Leider wird diese Funktion nicht mehr unterstützt, da bei den Linux Kerneln die für den Einsatz in symmetrischen Multiprozessorsystemen (SMP) konzipiert sind Änderungen an der Prozessverwaltung des Betriebssystems vorgenommen wurden (ab Ver. 2.6.14) [27]. Darüber hinaus können Probleme bei der Zuordnung von Kindprozessen auftreten, da diese eine andere PID als der Elternprozess besitzen.

### 4.1.4 UID matching

*Iptables* bietet neben den bereits genannten Parametern zusätzlich die Möglichkeit, Pakete auf den zugehörigen Benutzer zu prüfen. So können Prozesse unter verschiedenen Benutzer IDs (UID) gestartet und deren Pakete mit der jeweiligen UID gelabelt werden. Ähnlich zu dem Ansatz Prozesse einer „control group“ zuzuordnen, werden bei diesem Ansatz Prozesse einem Benutzer zugeordnet.
Unter Berücksichtigung der weiteren Ergebnisse wurde diese Methode als beste Lösung für das Problem der Zuordnung von Paket und zugehörigem Prozess ausgewählt. Begründung für diese Entscheidung ist die Kompatibilität mit der eingesetzten Software. Alle involvierten Programme sind mit der genutzten Betriebssystemversion (Ubuntu 14.04) kompatibel. Zudem kann bei ausreichender Anzahl von Benutzerkonten auf dem System eine detaillierte Zuordnung zu einzelnen Prozessen erfolgen.
Die Vorgehensweise bei diesem Ansatz besteht darin, eine Reihe von Benutzerkonten auf dem Rechner zu erstellen. Jeder gestartete Prozess wird dann genau einem Benutzer zugeordnet. Die Ausführung von einem Prozess pro Benutzer gewährleistet, dass alle Pakete, die von diesem Benutzer versendet und empfangen werden, zu dem zugeordneten Prozess gehören. Um die endgültige Zuordnung von Paketen zum Prozess zu ermöglichen, muss zuvor festgehalten werden, welcher Prozess von welchem Benutzer ausgeführt wird. Wie bereits bei den *cgroups* besteht auch bei diesem Ansatz ein Vorteil darin, dass alle Kindprozesse erfasst werden, da sie die UID des Elternprozesses erben und so ebenfalls dem selben Benutzer zugeordnet sind. Daraufhin werden in *iptables* die für die Zuordnung benötigten Regeln angelegt. Dies geschieht analog zu der Vorgehensweise beim Prüfen auf *cgroups*. Nachfolgend werden die Pakete für die einzelnen Nutzer mithilfe verschiedener *nflog*-Gruppen aufgezeichnet. Diese Daten werden daraufhin in der Nachbearbeitung dazu verwendet, den Schlüssel für den Datensatz zu erzeugen, indem jedem Paket eine UID und jeder UID ein Prozess zugeordnet werden.

## 4.2 Umsetzung

Im Folgenden wird die Umsetzung zur Erzeugung der Angriffe und des Hintergrundverkehrs in den jeweiligen Modulen mittels Shell-Skripten vorgestellt. Dies geschieht exemplarisch am Beispiel jeweils eines Angriffs und eines Typs von Hintergrundverkehr. Shell-Skripte fassen mehrere Kommandozeilenbefehle zusammen. Bei Aufruf eines Skriptes können komplexe Aktionen mit einer Vielzahl von Einzelbefehlen mit nur einem Aufruf ausgeführt werden. Sie dienen häufig der Arbeitserleichterung und ersparen das jeweilige explizite Aufrufen der Einzelbefehle durch den Benutzer. Shell-Skripte können ebenso wie Befehle mit Parametern aufgerufen werden. Da im Rahmen der Implementierung viele Aufrufe bereits vorhandener Programme erfolgen, wurde die Nutzung von Shell-Skripten gewählt.
Bevor die Erzeugung eines Datensatzes beginnen kann, muss der Benutzer zunächst einige Parameter in der Steuerungskomponente festlegen. Diese sind:

- der Name des Datensatzes [Experimentname]
- Dauer
- Auswahl von Angriffen
- Auswahl von Hintergrundverkehr

Nach Eingabe der Parameter beginnt die Steuerungskomponente damit, die Rechner für die Aufzeichnung des Datensatzes vorzubereiten (Abb. 4.3). Dazu wird zunächst ein neues Verzeichnis für das Experiment mit dessen Namen angelegt. In diesem Ordner werden später die gesammelten Daten zusammengefasst. Darüber hinaus werden mithilfe des Skripts *user_anlegen.sh* die zusätzlichen Benutzerkonten angelegt, die für die Ausführung der Prozesse und das Logging benötigt werden. Falls die Benutzerkonten nicht dauerhaft erzeugt werden sollen, muss am Ende des Experiments das Skript *user_loeschen.sh* aufgerufen werden.
Abbildung 4.3 beinhaltet einen Teilabschnitt der Testumgebung (Abb. 3.2). Es wird die beispielhafte Umsetzung eines Angriffs wie auch eines Typs von Hintergrundverkehr dargestellt, um die Funktionsweise des Frameworks zu verdeutlichen. Neben den Parametern werden die Aufgaben der Steuerungskomponente aufgeführt. Für die beiden involvierten Rechner werden jeweils die Skripte angeführt, die durch die Steuerungskomponente zu platzieren und auszuführen sind. Die dargestellten Beispiele eines Typs von Angriff und Hintergrundverkehr werden im Folgenden detailliert ausgeführt.

### 4.2.1 Angriffe

Für den zu exemplarisch implementierenden Angriff wurde ein Brute-Force-Angriff ausgewählt.
Basis dieses Angriffs bildet das Programm *THC-Hydra* [28]. Dieses wurde ausgewählt, da es Angriffe auf eine Vielzahl verschiedener Protokolle zulässt. Unter diesen befinden sich viele populäre Dienste wie *SSH*, *FTP*, *HTTP*, *SVN* oder *SMB*. Entwickelt und unterstützt wird das Programm von einer internationalen Gruppe von Forschern und Hackern mit dem Namen „The Hacker's Choice“ (THC). Es dient dem Testen von Rechnernetzen durch Systemadministratoren, um Sicherheitslücken zu finden. Der unsachgemäße Einsatz außerhalb eines dafür vorgesehenen Netzes, beispielsweise durch Angriffe gegen öffentliche Ziele, ist strafbar. Im Rahmen dieser Arbeit wurde innerhalb der Testumgebung ein Beispielangriff auf das *SSH*-Protokoll durchgeführt. Zu diesem Zweck müssen zunächst einige Vorbereitungen getroffen werden.
Zuerst muss das Programm *THC-Hydra* auf dem Rechner, der Ausgangspunkt des Angriffs sein soll, installiert werden. Die Installation von *THC-Hydra* erfolgt aus dem Paketquellen der genutzten Ubuntu Distribution mithilfe des „Advanced Packaging Tool“ (apt). Danach müssen die Listen, welche die zu testenden Logins sowie Passwörter enthalten, durch die Steuerungskomponente platziert werden. Als Verzeichnis für die

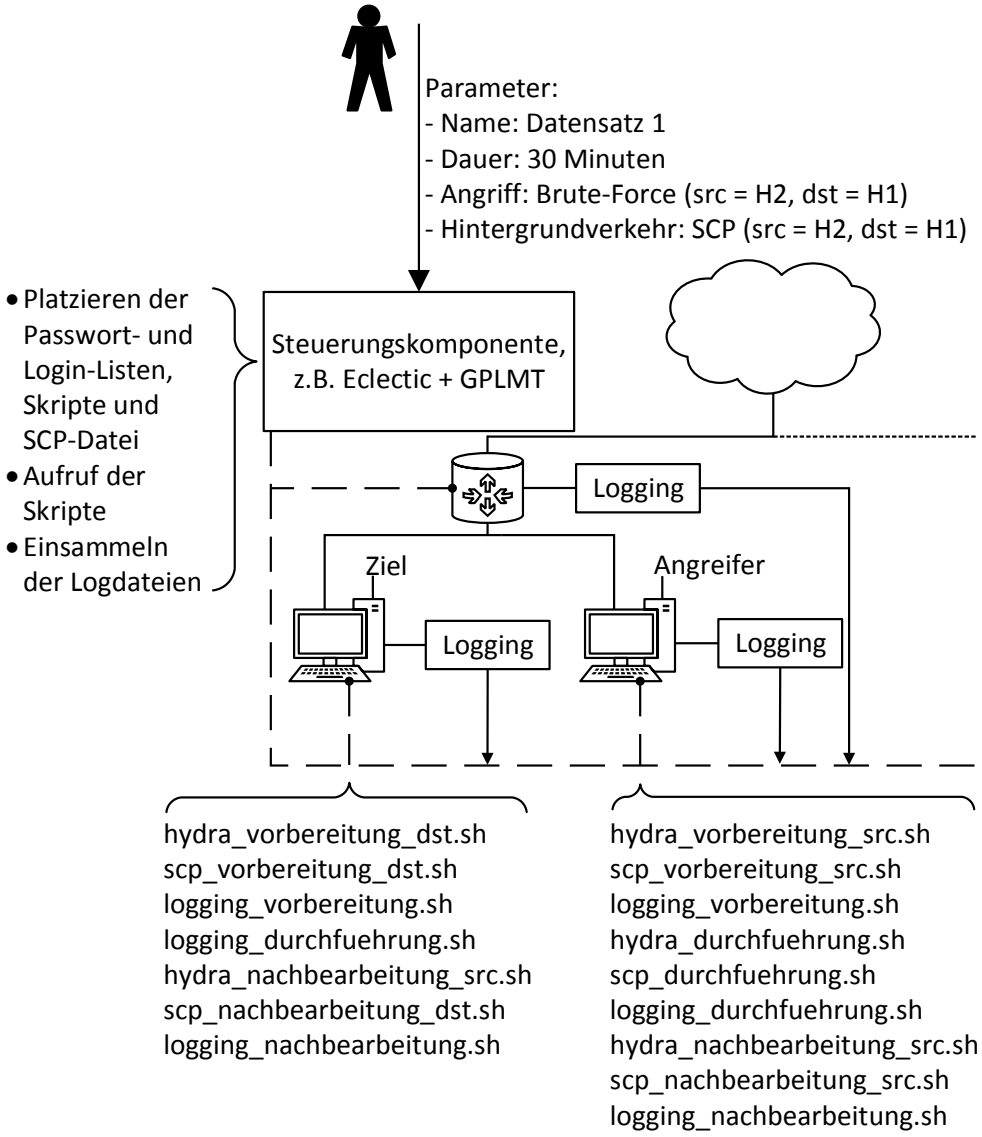


Abbildung 4.3: Framework Funktionsweise

Dateien wird ein für das Experiment angelegtes Verzeichnis gewählt, dessen Inhalt später durch die Steuerungskomponente eingesammelt wird. Parallel zu den Vorbereitungen auf dem Angriffsrechner muss auf dem Zielrechner der *SSH*-Server installiert werden. Während der Tests wurde der *openssh-server* verwendet, der ebenfalls in den Paketquellen von Ubuntu vorhanden ist und mithilfe des apt installiert wird.
Diese Schritte wurden mittels zweier Shell-Scripts umgesetzt. Die Datei *hydra_vorbereitung_src.sh* muss durch die Steuerungskomponente auf den Quellrechner platziert und aufgerufen werden. Die Datei *hydra_vorbereitung_dst.sh* ist für die Ausführung auf dem

Zielrechner vorgesehen.
Die Liste mit den zu testenden Logins wird bereits vor dem Experiment erzeugt.
Als zu testende Passwörter wurde die „Rockyou“ Passwortliste gewählt. Diese umfasst mehr als 14 Millionen Passwörter, die aus realen Passwortdiebstählen resultieren. Zudem sind die Passwörter in absteigender Reihenfolge nach der Häufigkeit ihres Vorkommens sortiert. Diese Kriterien lassen den Schluss zu, dass diese Liste auch in realen Angriffen genutzt wird, welche während des Experiments möglichst genau modelliert werden sollen.
Nachdem diese Vorbereitungen getroffen wurden, wird im Hauptteil während der Durchführung des Experiments *THC-Hydra* mit dem Befehl

```
hydra -L loginpath -P pwpath dstip service
```

aufgerufen. Die genutzten Parameter sind *-L* zum Aufruf einer Loginliste, *-P* zum Aufruf einer Passwortliste, die Ziel-IP-Adresse sowie der zu attackierende Dienst. Dieser Aufruf befindet sich im Skript *hydra_durchfuehrung.sh*. Der Aufruf des Skripts erfolgt nach dem Schema:

```
./hydra_durchfuehrung.sh [Experimentname] [ZielIP] [Dienst] [Loginliste] [Passwortliste]
    ↪ [Wartezeit]
```

Dem Skript müssen die Parameter [Experimentname] und [ZielIP] frei wählbar übergeben werden. Mit dem optionalen Parameter [Wartezeit] kann die verzögerte Ausführung festgelegt werden. Der Parameter für den [Dienst] muss *ssh* lauten. Die Pfade für die Listen müssen denen aus der Vorbereitung entsprechen.
Im Rahmen der Nachbearbeitung müssen die benötigten Programme wieder deinstalliert werden. Das Skript *hydra_nachbearbeitung_src.sh* wird auf dem Quellrechner ausgeführt und deinstalliert *THC-Hydra*. Das Skript *hydra_nachbearbeitung_dst.sh* entfernt den *openssh-server* vom Zielrechner. Die Skripte, Listen sowie die während des Experiments erzeugte Logdatei mit allen Vorkommnissen und eventuell gefundenen Logins, werden im Rahmen der Zusammenführung der Ergebnisse von der Steuerungskomponente entfernt.

### 4.2.2 Hintergrundverkehr

Für die Erzeugung von Hintergrundverkehr wurde ebenfalls ein beispielhafter Dienst implementiert. Es wurde das Verschieben einer Datei zwischen zwei Rechnern mittels „Secure CoPy“ (SCP) umgesetzt.
SCP ist ein Programm zur sicheren Übertragung von Dateien zwischen zwei Rechnern. Für Authentifikation und Übertragung nutzt es das *SSH*-Protokoll.
Da SCP das *SSH*-Protokoll verwendet, muss auf den Zielrechner ein *SSH*-Server installiert werden. Wie bereits beim Angriffsmodul erfolgt die Installation des *openssh-server* ebenfalls mittels des apt. Durch die Steuerungskomponente müssen Dateien für den Download auf dem Zielrechner platziert werden. Die Vorbereitung wurde durch das Skript *scp_vorbereitung_dst.sh* implementiert. Auf dem Quellrechner muss das Werkzeug

*sshpass* installiert werden. Dies wird zur Eingabe des Benutzerpassworts für die *SSH*-Authentifikation während der Durchführung benötigt. Die Installation erfolgt wiederum aus den Paketquellen und ist in der Datei *scp_vorbereitung_src.sh* implementiert.
Während der Durchführung wird dann SCP gestartet. Da bei der *SSH*-Verbindung die Authentifizierung mittels Benutzername und Passwort erfolgt, muss das Werkzeug *sshpass* verwendet werden, um die verzögerte Passwortabfrage zu umgehen und das Passwort innerhalb des Skripts direkt einzugeben. Dies ist durchaus sicherheitskritisch, da das Passwort im Klartext als Parameter vorkommt. Da alle Aktionen jedoch in einer abgeschlossenen Testumgebung stattfinden, deren Rechner keine sensiblen Daten enthalten, wird der Sicherheitsaspekt hier vernachlässigt. Um auf dem Quellrechner keinen Speicherplatz mit der übertragenen Datei zu belegen wird diese direkt verworfen, indem sie nach */dev/null* geschrieben wird. Der Aufruf des Skripts erfolgt nach dem Schema:

```
./scp_durchfuehrung.sh [ZielIP] [Benutzername] [Passwort] [Dateipfad] [Wartezeit]
```

Die [ZielIP] gibt die Adresse des *SSH*-Servers an. Der [Benutzername] ist der Name, unter dem sich mithilfe des [Passworts] eingeloggt werden soll, um die Datei unter dem angegebenen [Dateipfad] zu übertragen. Die [Wartezeit] gibt wiederum die optionale Verzögerung an, mit der die Aktion erfolgen soll.
In der Nachbearbeitungsphase wird mithilfe des *scp_nachbearbeitung_src.sh* Skripts *sshpass* auf dem Quellrechner deinstalliert. Zur Deinstallation des *openssh-server* muss auf dem Zielrechner die Datei *scp_nachbearbeitung_dst.sh* ausgeführt werden. Die zu übertragende Datei wird von der Steuerungskomponente vom Zielrechner entfernt.

### 4.2.3 Logging

Das Aufzeichnen der Datensätze soll mittels des in 4.1.4 beschrieben Verfahrens erfolgen. Die Umsetzung erfolgt ebenfalls mithilfe von Shell-Skripten für die verschiedenen Abschnitte. Das Logging findet auf jedem Rechner statt, der am Experiment beteiligt ist (Abb. 4.3).
Zunächst wird das Programm *ulogd2* installiert. Dieses wird zur Aufzeichnung der Pakete aus den einzelnen *nflog*-Gruppen benötigt. Ursprünglich sollte dies mithilfe von *Tcpdump* oder *Dumpcap* erfolgen. Nach Komplikationen, die in 4.2.5 dargelegt werden, wurde *ulogd2* als Alternative ausgewählt. Im Gegensatz zu *Tcpdump* und *Dumpcap* werden bei *ulogd2* die Konfigurationen für das Logging mittels einer Datei vorgenommen. Diese muss ebenfalls zu Beginn in das Verzeichnis von *ulogd2* übertragen werden. In der Konfiguration von *ulogd2* muss darauf geachtet werden, dass mindestens so viele *nflog*-Gruppen aufgezeichnet werden, wie Benutzerkonten auf dem Rechner vorhanden sind. Sonst könnten später Gruppen erzeugt werden, die nicht aufgezeichnet werden. Zusätzlich muss der Maximalwert des Empfangspuffers erhöht werden, damit der in 4.2.5 beschriebene Pufferüberlauf vermieden wird. Dazu muss die Zeile *net.core.rmem_max = [Wert]* an die Datei */etc/sysctl.conf* angefügt werden. Der [Wert] darf maximal *2147483647*

B betragen. Mit dem Befehl *sudo sysctl -p* wird der neue Wert daraufhin geladen. Während der Vorbereitung des Loggings werden des Weiteren alle Benutzer des Systems ausgelesen. Die zusätzlich für das Logging benötigten Benutzerkonten wurden zuvor durch die Steuerungskomponente angelegt. Die Namen werden dann zeilenweise mit zugehöriger UID in die Datei *UID-Name.txt* gespeichert. Diese Datei wird im nächsten Schritt zeilenweise eingelesen und für jeden Eintrag (Username, UID) werden die drei für das Logging benötigten Regeln zu *iptables* hinzugefügt. Diese sind:

```
sudo iptables -A OUTPUT -m owner --uid-owner userid -j CONNMARK --set-mark value
```

Diese Regel benutzt das „owner matching“ von *iptables* und prüft Pakete in der *OUTPUT*-Kette auf den Benutzers mit der UID *userid*. Bei Zutreffen wird *CONNMARK* damit beauftragt, sämtliche Pakete mit den angegebenen Wert *value* zu markieren. Wichtig ist, dass auch eingehende Pakete der Verbindungen mit dem Wert *value* markiert werden, da *CONNMARK* ganze Verbindungen markiert.

```
sudo iptables -A INPUT  -m connmark --mark value -j NFLOG --nflog-group nlgroup
```

Mit dieser Regel werden alle an den Rechner adressierten Pakete in der *INPUT*-Kette, die zuvor mit dem Wert *value* markiert wurden, in die *nflog*-Gruppe *nlgroup* weitergeleitet.

```
sudo iptables -A OUTPUT -m connmark --mark value -j NFLOG --nflog-group nlgroup
```

Diese Regel führt die selbe Aktion aus wie die Regel zuvor, dieses Mal jedoch für vom Rechner ausgehende Pakete in der *OUTPUT*-Kette.
Die Werte der *nlgroup* sowie des *value* müssen für jeden Benutzer unterschiedlich sein, da sonst Pakete mehrerer Benutzer gemischt werden. Die Vorbereitung des Loggings werden durch das Skript *logging_vorbereitung.sh* eingeleitet.
Während der Durchführung werden dann die für das Logging verantwortlichen Prozesse mithilfe des Skripts *logging_durchfuehrung.sh* aufgerufen. Zuerst wird *Tcpdump* zur Aufzeichnung des gesamten ein- und ausgehenden Verkehrs gestartet.

```
sudo tcpdump -n -q -i eth0 -B 102400 -w [Pfad der Ausgabe]
```

Die genutzten Parameter sind *-n*, da eine Umwandlung von IP-Adressen in Namen nicht nötig ist und Performanz kosten kann, *-q* um unnötige Ausgaben von *Tcpdump* zu vermeiden und *-B* um den Empfangspuffer von *Tcpdump* zu vergrößern. Bei hohen Datenraten kann so ein Pufferüberlauf vermieden werden. Im Beispiel wurden 100 MB gewählt. Mittels des *-i* Parameters wird das Interface angegeben, von dem aufgezeichnet werden soll und mit *-w* wird die Ausgabe spezifiziert.
Danach wird *ulogd2* gestartet, das den Netzwerkverkehr jedes Benutzers in eine separate *.pcap*-Datei aufzeichnet. Da alle Einstellungen bereits in der Konfigurationsdatei erfolgt sind, werden keine weiteren Parameter zum Start von *ulogd2* benötigt.
Das für die Nachbearbeitung verantwortliche Skript *logging_nachbearbeitung.sh* umfasst das Beenden von *ulogd2* sowie *Tcpdump*. Um den Rechner in den Ausgangszustand zurückzuversetzen, muss *ulogd2* deinstalliert und der Wert für die Maximalgröße des Empfangspuffers wieder auf den ursprünglichen Wert zurückgesetzt werden.

### 4.2.4 Schlüsselerzeugung

Im Anschluss an die Aufzeichnung der Pakete muss die Zuordnung von Paketen und Benutzern erfolgen. Zu diesem Zweck werden die *.pcap*-Dateien für die einzelnen Benutzer sowie der Mitschnitt des gesamten Netzwerkverkehrs benötigt.
Die Implementierung der Schlüsselerzeugung erfolgt in Java mithilfe der *jNetPcap*-Bibliothek. Ziel ist die Erstellung einer Liste, die jedem Paket in der Gesamt-Aufzeichnung die zugehörige UID zuordnet.
Für das Labeling der Netzwerkpakete wird zunächst der Zusammenhang zwischen der Aufzeichnung des gesamten Netzwerkverkehrs und dem der einzelnen Benutzer betrachtet. Da die Pakete sämtliche Benutzer des Rechners aufgezeichnet werden und jeder Prozess einem Benutzer zugeordnet ist, müssen Aufzeichnungen der einzelnen Benutzer in der Summe dem gesamten Netzwerkverkehr entsprechen. Umgekehrt bedeutet dies auch, dass jedes Paket der Gesamt-Aufzeichnung in einer der Benutzer-Aufzeichnungen enthalten sein muss. Dieser Zusammenhang besteht jedoch nur unter der Annahme, dass die Pakete aller Benutzer, einschließlich Systemdiensten mit eigenem Benutzerkonto, aufgezeichnet werden und keine Pakete verloren gehen.
Die Vorgehensweise während des Labelings besteht darin, den Paketen der Gesamt-Aufzeichnung die entsprechenden Pakete in den Benutzer-Aufzeichnungen zuzuordnen (Abb. 4.4). Zu diesem Zweck wird zunächst die erste Benutzer-Aufzeichnung geöffnet. Daraufhin wird fortlaufend für jedes Paket die Gesamt-Aufzeichnung nach dem entsprechenden Paket durchsucht. Bei übereinstimmen der IP-Kopfdaten (IP-Header) und den IP-Nutzdaten (IP-Payload) wird dem Index des Pakets aus der Gesamt-Aufzeichnung der Name der Benutzer-Aufzeichnung, in der sich das entsprechende Paket befindet, zugeordnet. Wurde das letzte Paket der Benutzer-Aufzeichnung erreicht, wird die nächste Benutzer-Aufzeichnung geladen. Dies geschieht solange, bis alle Benutzer-Aufzeichnungen abgearbeitet wurden.
Die Erhaltung der Reihenfolge kann bei diesem Verfahren ausgenutzt werden. Wurde ein Paket aus der Benutzer-Aufzeichnung zugeordnet, kann die Suche des nächsten Pakets an der aktuellen Stelle in der Gesamt-Aufzeichnung fortgesetzt werden, da das zu findende Paket nicht vor dem zuvor zugeordnetem Paket liegen kann (Abb. 4.4).
Während der Zuordnung können zwei Fehler auftreten. Beim ersten Fehler enthält die Gesamt-Aufzeichnung ein Paket, das in der Benutzer-Aufzeichnung fehlt. In diesem Fall würde das Paket nicht gelabelt werden. Eine weitere Fehlerbehandlung ist nicht möglich, da nicht bekannt ist, in welchem der Benutzer-Aufzeichnung das Paket fehlt. Beim zweiten Fehler enthält die Benutzer-Aufzeichnung ein Paket, das in der Gesamt-Aufzeichnung nicht vorhanden ist. In diesem Fall könnte das Paket nicht zugeordnet werden. Um den Fehler zu beheben müsste das fehlende Paket der Gesamt-Aufzeichnung hinzugefügt werden. Diesem Vorgehen liegt die Annahme zugrunde, dass alle Pakete der Benutzer-Aufzeichnung auch in der Gesamt-Aufzeichnung enthalten sein müssen.
Der Vergleich der Paketdaten erfolgt mithilfe von IP-Kopf- und -Nutzdaten, da die durch *nflog* erfassten Benutzer-Aufzeichnung keine Ethernet-Kopfdaten (Ethernet-Header)

enthalten. Daher ist ein Labeling von Paketen ohne IP-Kopfdaten nicht möglich. Diese Pakete sind im Schlüssel nicht enthalten.

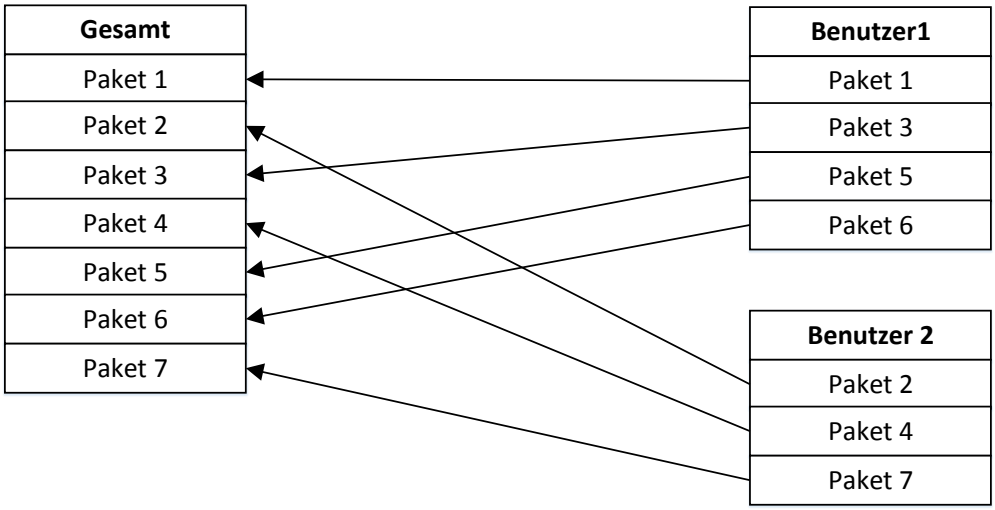


Abbildung 4.4: Labeling Funktionsweise

### 4.2.5 Probleme

Während der Implementierungen traten mehrere Fehler mit *Dumpcap* und *Tcpdump* auf. Diese werden im Folgenden ausführlicher behandelt.
Die ersten Tests der Logging-Komponente wurden innerhalb eines Rechnernetzes mit einer Übertragungsrate von 100 Mbit/s durchgeführt. Da jedoch in den Anforderungen die Annahme realistischer Übertragungsraten vorgeschrieben ist, wurde im Zuge der weiteren Tests in eine Umgebung mit Gigabit Anbindung gewechselt. Durch die weit höhere Übertragungsrate traten jedoch neue Probleme auf. Der Puffer, der für die eingehenden Pakete genutzt wird, reichte nicht mehr aus, sodass ein Pufferüberlauf und daraus folgend ein Fehler auftrat, nachdem *Dumpcap* sowie *Tcpdump* sich automatisch beendeten. Dieses konnte mithilfe des *-B* Parameters bei Programmaufruf nicht ohne weiteres behoben werden. Nach Recherche wurde die Systemvariable im Verzeichnis */proc/sys/net/core/rmem_max* identifiziert, welche den Maximalwert des Eingangspuffers limitiert. Nach Erhöhung auch dieser Variable konnte das Problem des sofortigen Pufferüberlaufs behoben werden. Bei fortschreitender Experimentlaufzeit trat jedoch immer noch ein Pufferüberlauf auf. Als Grund hierfür konnte die zu geringe Schreibgeschwindigkeit der Festplatte identifiziert werden. Durch das Schreiben der Paketdaten nach */dev/null* zu Testzwecken, trat kein weiterer Pufferüberlauf auf. Es konnten somit keine real Tests mit Aufzeichnungen in Dateien stattfinden. Da jedoch nach heutigem Standard Festplatten sowie Halbleiterlaufwerke (SSD) mit Schreibraten von weit über

100 MB/s existieren, kann das vorgestellte Verfahren in entsprechend performanten Systemen zur Anwendung kommen. Darüber hinaus handelt es sich bei den Testfällen um Extremsituationen, in denen die gesamte zur Verfügung stehende Netzwerkkapazität ausgelastet wird. Im Realfall könnte die Übertragungsrate demnach geringer sein, so dass keine Probleme mit dem Puffer mehr auftreten.
Um die Auslastung der Logging-Komponente zu reduzieren, kann das Logging auf die Erfassung des Gesamt-Verkehrs sowie des Angriffsverkehrs beschränkt werden. Da Angriffe in der Regel wenig Verkehr erzeugen, um verdeckt agieren zu können, werden die durch die Logging-Komponente zu erfassenden Daten reduziert. In diesem Fall können die Angriffspakete detailliert gelabelt werden. Die Pakete des Hintergrundverkehrs würden aufgrund der fehlenden Informationen jedoch nur noch als harmlos („nicht Angriff“), ohne die Angabe weiterer Details, gelabelt werden. Die Nutzung dieses Verfahrens würde jedoch trotzdem Datensätze erzeugen, die den Anforderungen entsprechen, da bei der Evaluation von IDS vor allem die Details der Angriffspakete von Bedeutung sind.
Nach Lösung des Pufferüberlaufs trat auch weiterhin eine Fehlermeldung auf, die zur Beendigung von *Dumpcap/Tcpdump* führte. Laut Fehlerbericht tritt ein abgeschnittenes Paket auf (Message truncated). Diese Fehlermeldung ereignet sich jedoch nur in Verbindung mit der Nutzung von *nflog*. Für diese Fehlermeldung konnte keine Lösung gefunden werden. Daher wurde nach alternativen Programmen für die Aufzeichnung des Netzverkehrs gesucht. Als weitere Möglichkeit wurde *ulogd2* gefunden. *ulogd2* ist ein Log-Daemon, der unter anderem Pakete aus *nflog*-Gruppen aufzeichnen kann. Mithilfe von *ulogd2* ist es ebenfalls möglich, Pakete einzelner *nflog*-Gruppen in separate *.pcap*-Dateien aufzuzeichnen. Bei *ulogd2* muss jedoch von Beginn an die Anzahl der aufzuzeichnenden *nflog*-Gruppen feststehen. Alle Einstellung für das jeweilige Experiment werden in den *ulogd2* Konfigurationsdatei getroffen, die beim Start des Programms geladen wird.

## 4.3 Eclectic

Wie in 3.2.4 vorgestellt, wird für die Überwachung des Experiments und die Ausführung der einzelnen Skripte eine Steuerungskomponente benötigt. Aufgrund der aufgetretenen Probleme konnte die implementierte Lösung nicht mehr in der Testumgebung umgesetzt werden. Diese bietet jedoch für die Ausführung von Befehlen und die Platzierung von Dateien auf den einzelnen Testrechnern und virtuellen Maschinen die Umsetzung durch die Eclectic Software Umgebung. Speziell das „GNUnet Parallel Large-scale Management Tool“ (GPLMT) Werkzeug bietet alle in 3.2.4 geforderten Funktionen an eine Steuerungskomponente [29].
Vor Beginn des Experiments können über den Reiter „*Experiments*“ die Skripte auf die einzelnen Rechner übertragen werden. Dazu wird das zu übertragene Skript ausgewählt und aus einer Liste aller Rechner können diejenigen selektiert werden, auf die

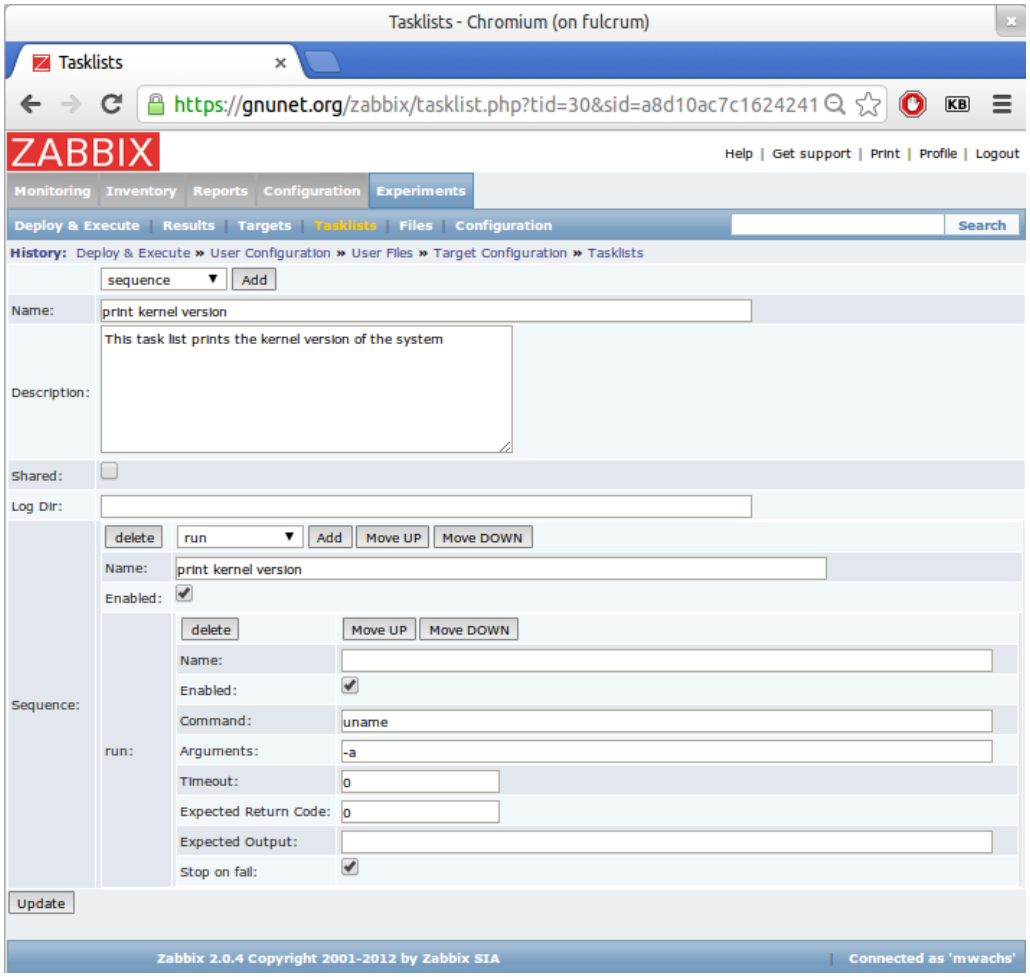


Abbildung 4.5: GPLMT Oberfläche [29]

das Skript übertragen werden soll. Nach diesem Schema werden auch alle weiteren benötigten Dateien auf die Testrechner übertragen. Diese Schritte können zusammengefasst und automatisch ausgeführt werden, so dass lediglich einmal der Auftrag erteilt werden muss, alle Dateien zu platzieren. GPLMT ermöglicht neben der Ausführung von Skripten sämtliche Funktionen der Kommandozeile zu nutzen, so dass Übergangs- und Vorbereitungsoperationen zwischen den einzelnen Skripten vereinfacht werden [29].
Zur Ausführung eines Experiments muss beispielsweise zunächst eine neue „*Tasklist*“ pro Host erzeugt werden. Diese besteht aus einem Einzelnen oder einer Sequenz von Skripten, wie in Abbildung 4.3 dargestellt wird. Für jedes einzelne Skript muss dieses als *Command* sowie dessen Parameter als *Arguments* übergeben werden. Zudem sind weitere Funktionen wie der Abbruch der Sequenz bei Fehlschlag eines Befehls sowie die Angabe erwarteter Ausgaben möglich (Abb. 4.3).
Im Übergang zur Durchführung werden dann alle Skripte zum Starten des Loggings so-

wie der Hintergrundverkehrserzeugung und der Angriffe ausgeführt. Diese können, wie beschrieben, innerhalb einer Task-Liste gruppiert werden. Wie gefordert lässt GPLMT auch eine verzögerte Ausführung von Skripten sowie einen mehrfachen Aufruf zu.
Für die Nachbearbeitung muss ebenfalls eine eigene Task-Liste erstellt werden.
Nach Fertigstellung der einzelnen Komponenten ist somit der nächste Schritt, diese in die Eclectic Umgebung einzubinden, um das GPLMT als Steuerungskomponente zu nutzen.

# Kapitel 5

# Evaluation

Dieses Kapitel umfasst die Evaluierung des implementierten Frameworks. Die Qualität wird anhand der in 2.2 formulierten Anforderungen ermittelt. Im Verlauf der qualitativen Analyse wird festgestellt, ob das Framework alle Kriterien erfüllt, um Datensätze zu erzeugen, die den Anforderungen entsprechen.
Da das Logging ein essentiell wichtiger Bestandteil des Frameworks ist, wird zusätzlich eine Performanz-Analyse dieser Komponente durchgeführt. In diesem Teil wird überprüft, wie sich die Auslastung des Systems während des Loggings im Gegensatz zur Ausführung ohne Logging verhält. Die Auslastung mit aktiviertem Logging darf nicht zu hoch sein, um Einschränkungen im normalen Betriebsablauf zu vermeiden und die Pakete in Echtzeit aufzeichnen zu können.

## 5.1 Qualitative Analyse

*A1 Dokumentation*—Erster Punkt der Anforderungen ist die ausführliche Dokumentation. Die ausreichende Dokumentation der Implementierung wird durch die detaillierte Aufarbeitung innerhalb dieser Arbeit sichergestellt. Zusätzlich zu den Ausführungen in dieser Arbeit werden auch alle implementierten Teile mitgeliefert.
In 2.2 wird vom Labeling gefordert, dass es zu jedem Paket den zugehörigen Rechner, Prozess und die Aufgabe des Prozesses beinhaltet. Dies kann sichergestellt werden, da das Logging auf allen Rechnern stattfindet und beim späteren Labeling-Prozess leicht ermittelt werden kann, welcher Rechner welche Pakete verschickt hat. Durch den in 4.1.4 vorgestellten Ansatz für das detaillierte Logging ist ebenfalls die Angabe des Prozesses möglich. Da eine Zuordnung zum erzeugenden Prozess möglich ist, können Angriffsprozessen zweifelsfrei ihre Pakete zugeordnet werden. Somit ist die Information, ob ein Paket bösartig ist, ebenfalls inbegriffen.

*A2: Realistische Annahmen*—Durch die freie Wahl der Netzwerktopologie kann diese Anforderung erfüllt werden. Die zu Testzwecken erstellten Datensätzen wurden in einer kleinen Testumgebung aufgezeichnet. Das Framework ist aber nicht an eine bestimmte

Topologie gebunden, so dass problemlos auch Datensätze in größeren Rechnernetzen erstellt werden können.
Die Anforderung an eine angemessen hohe Datenrate wird erfüllt. Die während der Implementierung genutzte Datenrate entspricht 1000 Mbit/s. Diese stellt im Moment die schnellste Datenrate dar, die ohne spezielle Hardware erreichbar ist.
Durch den modularen Aufbau und die Erweiterbarkeit der Angriffskomponente können jederzeit neue Angriffe hinzugefügt werden. Somit kann die Anforderung an eine vielfältige Auswahl von Angriffen erfüllt werden. Durch die Aufzeichnung von Netzwerkverkehr innerhalb des Rechnernetzes können zudem aktuelle Angriffsszenarien, wie die in 2.1 vorgestellten, erfasst werden.
Durch die Flexibilität des Frameworks ist auch die Hintergrundverkehrskomponente auf neue Protokolle erweiterbar. Die Forderung nach Netzwerkverkehr, der eine Vielzahl verschiedener Protokolle enthält, kann so ebenfalls erfüllt werden.

*A3 Interaktion zwischen Angriffen und Hintergrundverkehr*—Da kein Traffic-Generator verwendet wird, sondern der Hintergrundverkehr durch das entsprechende Modul auf jedem Rechner direkt erzeugt wird, findet eine Interaktion mit den Angriffen statt.
Die Tabelle 5.1 fasst die Anforderungen zusammen. Deutlich erkennbar ist, dass alle Anforderungen an die Dokumentation (A1), realistischen Annahmen (A2) sowie die Interaktion zwischen Angriffs- und Hintergrundverkehr (A3) durch das Framework erfüllt werden.

Tabelle 5.1: Qualität Framework

| | Darpa 98 | KDD 99 | NSL-KDD 09 | Framework |
|---|---|---|---|---|
| A1 | - | X | X | ✓ |
| A2 | X | X | - | ✓ |
| A3 | X | X | X | ✓ |

## 5.2 Performanz-Analyse

In diesem Abschnitt wird der Fokus auf das Logging gelegt, da es ein elementarer Bestandteil des Frameworks ist. Es kann nicht wie Angriffe oder Hintergrundverkehr erweitert werden. In der Regel muss die Logging-Komponente bei grundlegenden Modifikationen komplett überarbeitet werden, da einzelne Teile voneinander abhängig sind. Wird beispielsweise die Art der Aufzeichnung geändert, können später Komplikationen beim Labeling der Pakete auftreten. Aus diesem Grund muss die ordnungsgemäße Funktion und die Performanz bereits während der Implementierung berücksichtigt und überprüft werden.
Im Folgenden wird zunächst der Versuchsaufbau und die Durchführung der Messungen beschrieben. Darauf folgt die Auswertung der Ergebnisse.

### 5.2.1 Aufbau und Durchführung

Um neben den erfüllten Anforderungen auch die Performanz der implementierten Lösung zu ermitteln, wurden Messungen durchgeführt, welche die Leistung und den Ressourcenbedarf des Logging-Moduls als zentraler Komponente aufzeigen. Zu diesem Zweck wurden drei verschiedene Parameter als Indikatoren für die Leistung der gewählten Lösung festgelegt.
Die Grundlage der Bewertung ist die Differenz zwischen der Auslastung eines Rechners ohne Logging, verglichen mit der Auslastung des Rechners, während die implementierte Logging-Komponente ausgeführt wird.
Der erste für die Bewertung der Performanz betrachtete Parameter ist die Auslastung des Hauptprozessors (CPU). Gewählt wurde dieser Wert, da bei hoher Auslastung der CPU Paketverluste auftreten können sowie die Ausführung weiterer Prozesse eingeschränkt wird. Der zweite erfasste Wert ist die Auslastung des Hauptspeichers. Gerade bei dem gleichzeitigen Logging vieler UIDs besteht die Möglichkeit, dass der Hauptspeicher stark ausgelastet wird. Dieser Wert ist wichtig, da bei hoher Auslastung die Auslagerung von Daten auf den Hintergrundspeicher erfolgt und das System in Folge dessen extrem an Performanz verliert. Der dritte erfasste Parameter ist die Netzwerkauslastung. Dieser Wert ist wichtig, da das Logging keine Auswirkungen auf die Übertragungsrate haben sollte. Auch bei steigender Anzahl der Benutzer sollte diese nicht auf einen zu niedrigen Wert absinken.

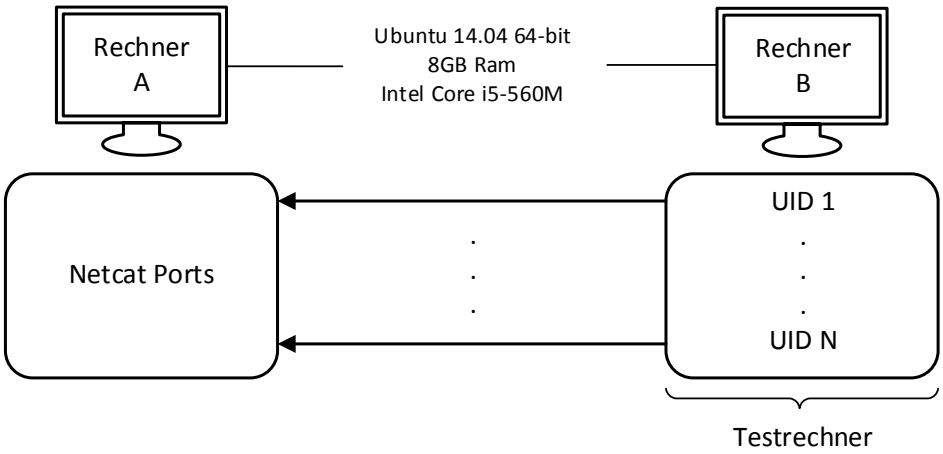


Abbildung 5.1: Testaufbau

Zur Erzeugung von Netzwerkverkehr wird *Netcat* verwendet. *Netcat* ist ein Werkzeug, mit dem Netzwerkverbindungen aufgebaut werden können. Über diese Verbindungen können Daten der Standardeingabe und -ausgabe versendet werden [30]. Um mithilfe

von *Netcat* Netzwerkverkehr zu erzeugen, werden auf einem entfernten Rechner 100 *Netcat* Instanzen gestartet, auf die der Testrechner Verbindungen aufbauen kann (Abb. 5.1). Ist eine Verbindung hergestellt, werden in einer unendlichen Folge Nullen an den Testrechner gesendet. Der Test beginnt damit, dass *Netcat* für einen Benutzer ausgeführt wird, worauf die Messdaten erfasst werden. Danach wird *Netcat* für einen weiteren Benutzer gestartet und eine zusätzliche Verbindung wird hergestellt. So werden Messdaten bis zu einer Anzahl von 100 Benutzern gesammelt. Diese Anzahl wird als ausreichend eingeschätzt, um ein detailliertes Logging durchführen zu können.
Um möglichst aussagekräftige Ergebnisse zu erhalten, wird die Logging-Komponente durch die komplette Ausnutzung der Übertragungsrate von 1000 Mbit/s maximal ausgelastet. Zudem wird die Anzahl der zu Loggenden UIDs stetig erhöht, so dass neben der Performanz zugleich das Verhalten der Komponente bei Extremlast untersucht wird. Bei realem Einsatz in Gigabit Umgebungen kann keine höhere Auslastung auftreten. Funktioniert die Komponente bei diesem Test, wird sie voraussichtlich auch später funktionieren.
Zur Erfassung der Messwerte werden zwei Programme verwendet. Zum einen ist dies *Collectl*, ein Programm zum Überwachen von System- und Ressourcenwerten, mit der Möglichkeit, die benötigten Werte zu erfassen. Während der Testläufe wird *Collectl* jeweils einmal pro neuem Benutzer aufgerufen und zeichnet für jeden zusätzlichen Benutzer die aktuellen Werte für CPU-, Hauptspeicher- und Netzwerkauslastung auf. Um Leistungsspitzen auszugleichen werden pro Benutzer jeweils 100 Messungen durchgeführt, von denen später das arithmetische Mittel gebildet wird. Das Zeitintervall zwischen den einzelnen Messungen beträgt 1 s. Zum anderen wird *Conky*, ebenfalls ein Werkzeug für die Erfassung von System- und Ressourcenwerten, benutzt. Im Gegensatz zu *Collectl* wird *Conky* einmal zu Beginn des Experiments gestartet und zeichnet Daten bis zum Ende des Testlaufs auf. Die Messwerte werden in einem Zeitintervall von 0.1 s erfasst.
Zur Ermittlung der Daten, mit und ohne Logging, werden zwei Testläufe gefahren. Um einen Referenzwert bei normaler Ausführung des Systems ohne Logging zu erhalten, wird zunächst ein Testlauf gestartet, bei dem nur die Programme *Conky* und *Collectl* für die Messungen sowie *Netcat* zur Erzeugung der Netzwerklast ausgeführt werden. Im zweiten Testlauf werden die Daten mit aktiviertem Logging erfasst. Der Test erfolgt nach dem selben Prinzip wie der Erste, es werden jedoch zu Beginn *Tcpdump* sowie *ulogd2* für 100 Benutzer gestartet. Bei jedem zusätzlichen Benutzer wird zusätzlich zu *Netcat* das Logging gestartet, indem die Regeln in *iptables* erzeugt werden. Da *ulogd2* nur Daten aus *nflog*-Gruppen aufzeichnet, erfolgt die Erfassung der Pakete erst nach Erstellung der Regeln, welche die Pakete in die *nflog*-Gruppen weiterleiten.

### 5.2.2 Ergebnisse

Betrachtet wurden die Parameter CPU-, Hauptspeicher- und Netzwerkauslastung. Für jeden dieser Faktoren enthalten die Abbildungen 5.2, 5.5 und 5.8 die durchschnittli-

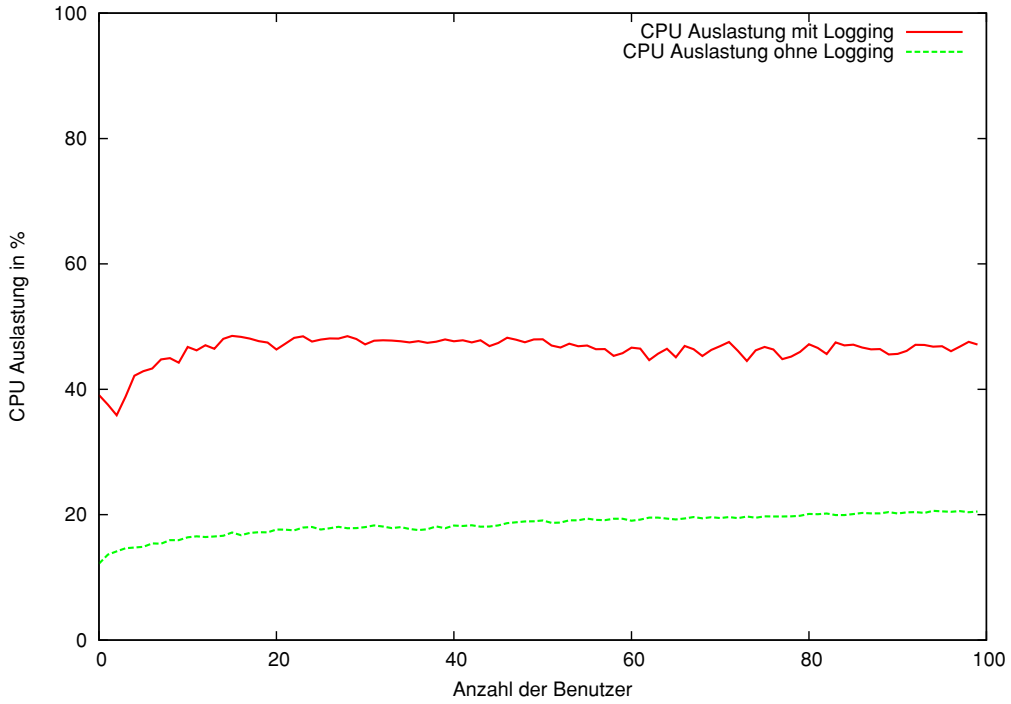


Abbildung 5.2: Auslastung CPU (*Collectl*)

chen Messwerte bei steigender Anzahl von Benutzern, die von *Collectl* erfasst wurden. Eingetragen sind jeweils zwei Graphen für die Messwerte mit und ohne Logging. Die Abbildungen 5.3, 5.6 und 5.9 beinhalten die Einzelmesswerte, die von *Conky* bei deaktiviertem Logging erfasst wurden. In den Abbildungen 5.4, 5.7 und 5.10 sind die Messwerte enthalten, die von *Conky* mit aktiviertem Logging ermittelt wurden. Die Werte an der Abszissenachse der durch *Conky* ermittelten Werte werden nicht in Benutzern, sondern in der Zahl der Messungen angegeben. Dies resultiert aus dem einmaligen Start von *Conky* zu Beginn und der fortlaufenden Messung bis zum Experimentende.
Für Differenzen zwischen den Messwerten mit und ohne Logging ist die Ausführung von *Tcpdump*, *ulogd2* und *iptables* verantwortlich.
Die Ergebnisse der CPU Auslastung zeigen eine um durchschnittlich etwa 27% erhöhte Auslastung bei eingeschaltetem Logging (Abb. 5.2). Demnach erzeugt das Logging einen deutlichen Zuwachs an Prozessorlast. Bei eingeschaltetem sowie ausgeschaltetem Logging zeigt der durch die zunehmende Benutzeranzahl bedingte Zuwachs bei der Auslastung des Hauptprozessors ein lineares verhalten. Der Graph bei eingeschaltetem Logging verläuft etwas ungleichmäßiger als der bei ausgeschaltetem Logging. Der allgemeine Verlauf beider Graphen deckt sich aber bis auf die Verschiebung an der Ordinatenachse. Dies bedeutet, dass das Logging einen konstanten Zuwachs an Last erzeugt. Im Rahmen der Leistungsbewertung ist das optimal, da auch bei höheren

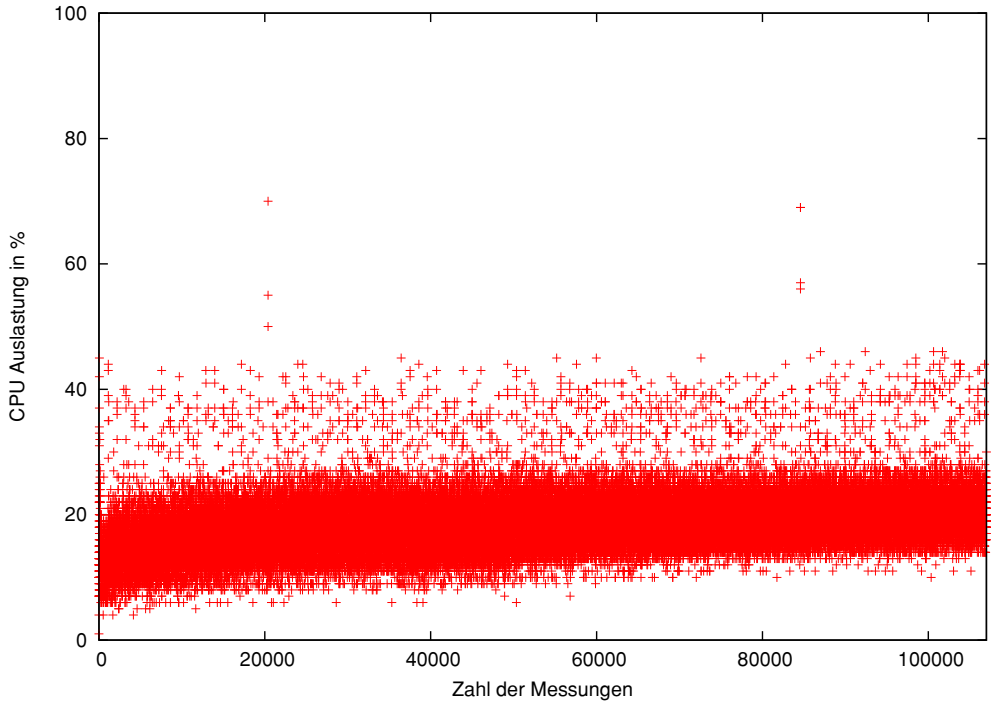


Abbildung 5.3: Auslastung CPU ohne Logging (*Conky*)

Benutzerzahlen die Auslastung nicht sprunghaft ansteigt. Da die Last mit eingeschaltetem Logging mindestens so stark ansteigen muss wie ohne Logging, kann an dieser Stelle keine weitere Optimierung erfolgen. Lediglich der konstante Faktor, um den die Last bei eingeschaltetem Logging höher ist, könnte optimiert werden. Da die maximale Auslastung aber gerade einmal 50% der Prozessorlast erreicht, ist dies im Rahmen dieser Arbeit nicht notwendig.

Die Messwerte der Hauptspeicherauslastung weisen ebenfalls darauf hin, dass der Zuwachs an Ressourcenbedarf bei steigender Anzahl von Benutzern linear verläuft (Abb. 5.5). Auch bei der Hauptspeicherauslastung erzeugt das Logging einen Zuwachs beim Ressourcenverbrauch um einen konstanten Faktor. Dieser beträgt zwischen beiden Testläufen etwa 200 000 KiB. In Bezug auf die Hauptspeicherauslastung ist der gewählte Ansatz also ebenfalls als optimal zu bezeichnen, zumal ein konstanter Zuwachs von 200 000 KiB für aktuelle Hauptspeichergrößen kaum ins Gewicht fällt.

Die Messwerte der Datenübertragungsraten zeigen ein unerwartetes Verhalten. Entgegen der zu erwartenden Annahme, dass die Übertragungsrate bei eingeschaltetem Logging sinkt, ist sie höher als bei deaktiviertem Logging (Abb. 5.8). Dieses Verhalten gab Anlass, die Testläufe zu wiederholen, da ein Fehler vermutet wurde. Jedoch trat auch bei mehrfacher Wiederholung dasselbe Muster auf. Eine Erklärung für dieses Phänomen konnte nicht gefunden werden. Es können jedoch höhere Werte der CPU- und Haupt-

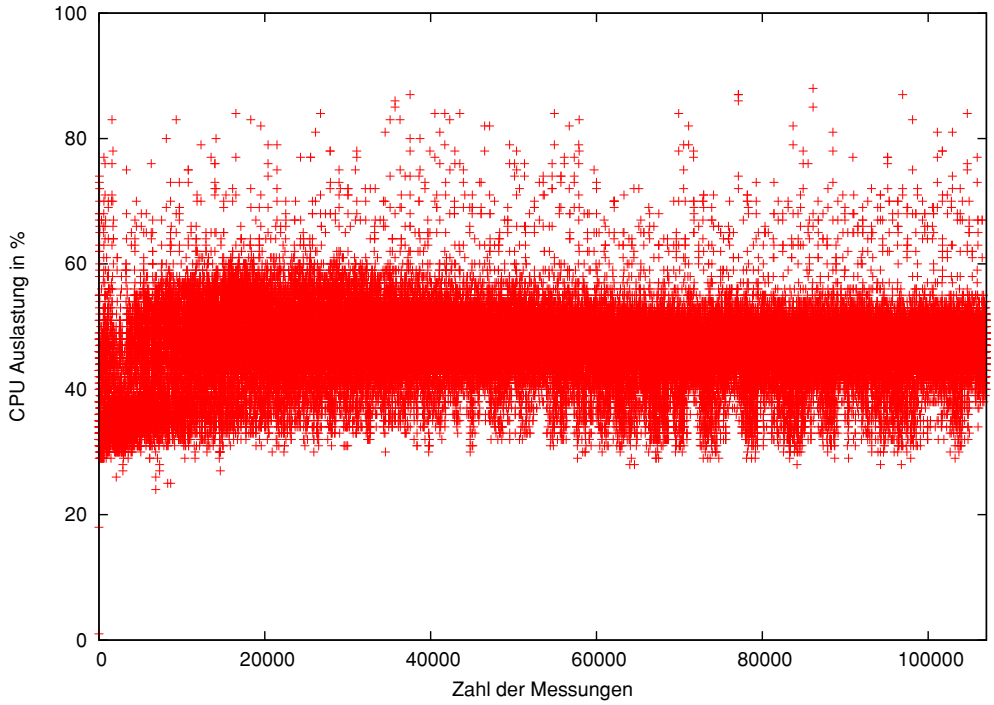


Abbildung 5.4: Auslastung CPU mit Logging (*Conky*)

speicherauslastung bei aktiviertem Logging hervorgerufen werden, da diese durch eine höhere Datenrate stärker ausgelastet werden. Die Werte dieser Ressourcen befinden sich jedoch in einem akzeptablen Bereich. Lediglich die im Vergleich zum deaktivierten Logging etwas stärker ansteigende Prozessorlast zu Beginn des Experiments könnte auf dieses Phänomen zurückzuführen sein.
Zusammenfassend lässt sich der implementierte Ansatz anhand der Messwerte als gut bewerten. Die zusätzlichen Auslastungen bei Prozessor und Hauptspeicher zeigen lediglich konstante Werte. Die Erhöhung der Übertragungsrate bei aktiviertem Logging weicht von den Erwartungen ab, lässt aber keine negativen Auswirkungen erkennen. Raum zur Optimierung bieten also lediglich die konstanten Zuwächse bei CPU und Hauptspeicherauslastung.

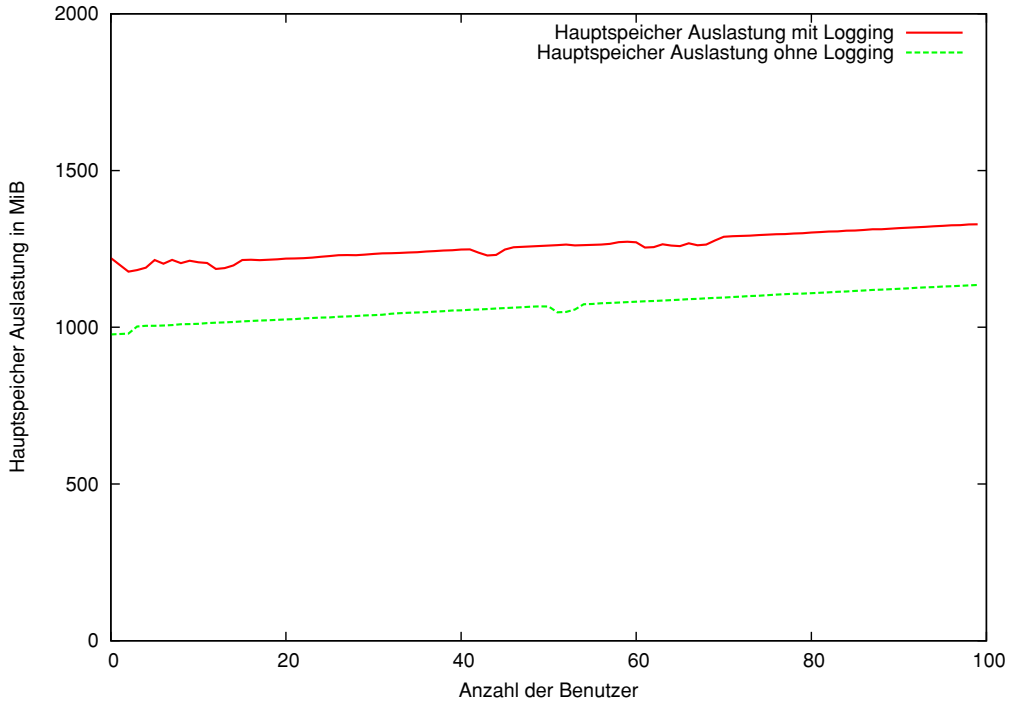


Abbildung 5.5: Auslastung Hauptspeicher (*Collectl*)

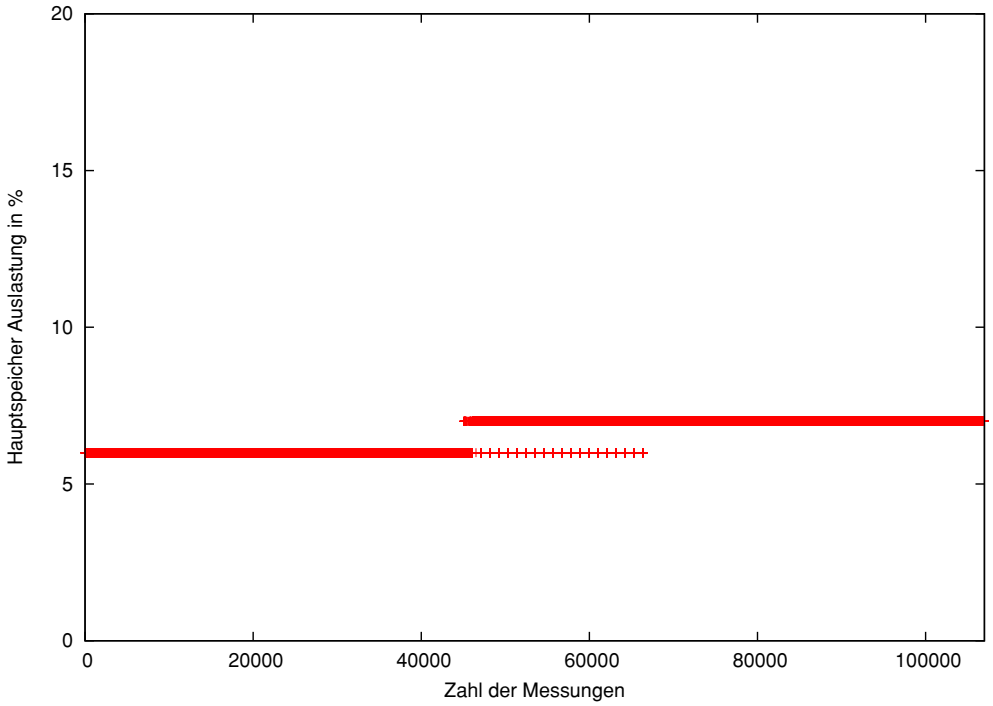


Abbildung 5.6: Auslastung Hauptspeicher ohne Logging (*Conky*)

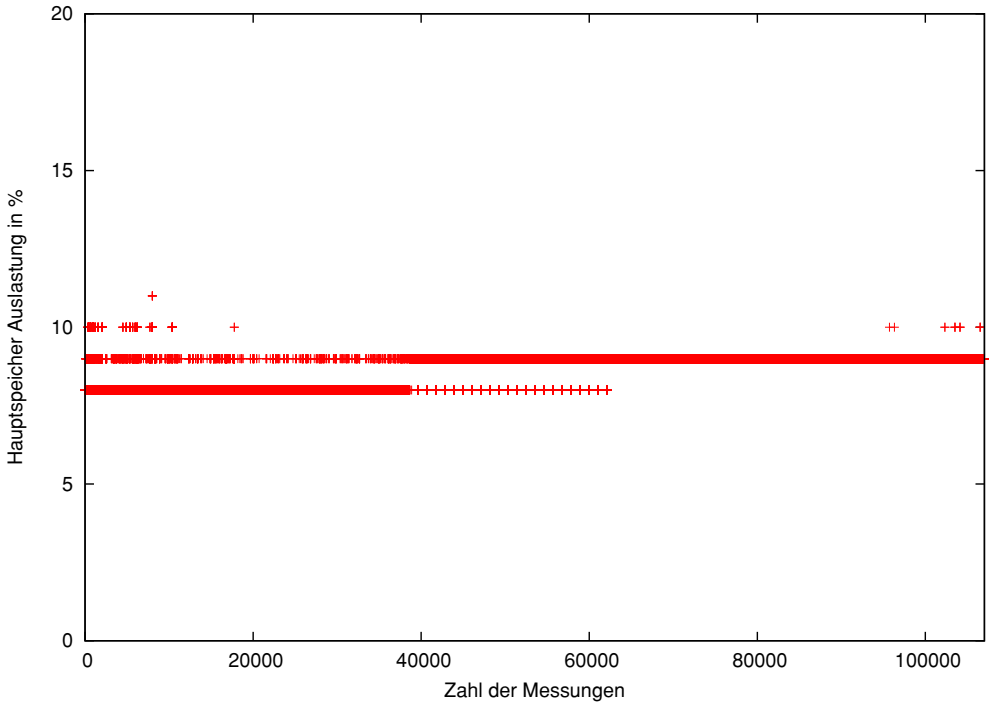


Abbildung 5.7: Auslastung Hauptspeicher mit Logging (*Conky*)

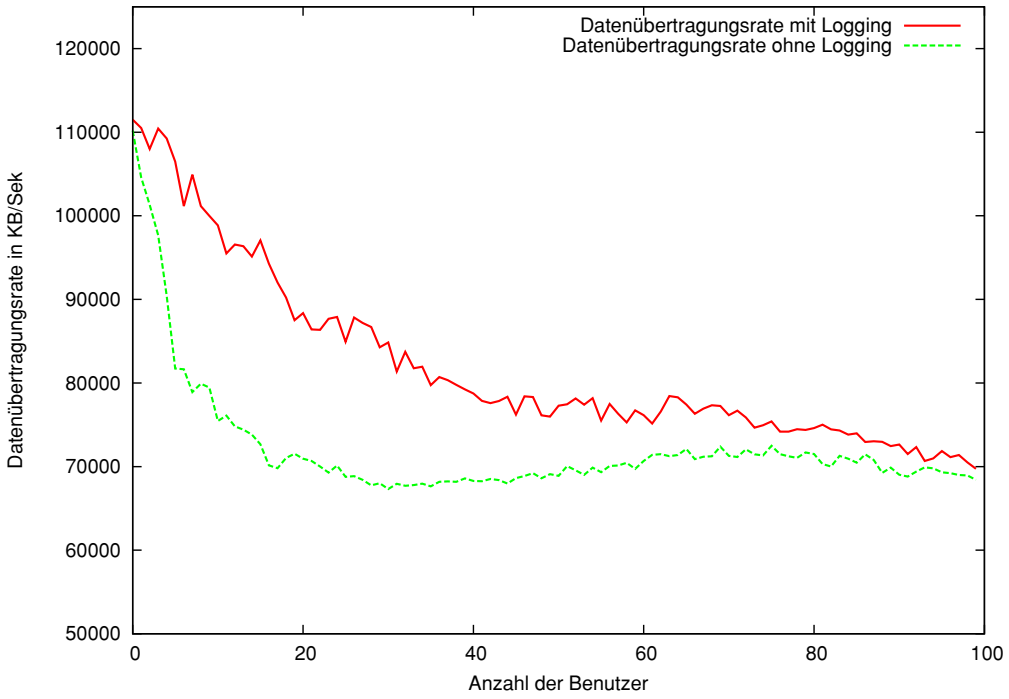


Abbildung 5.8: Auslastung Netzwerk (*Collectl*)

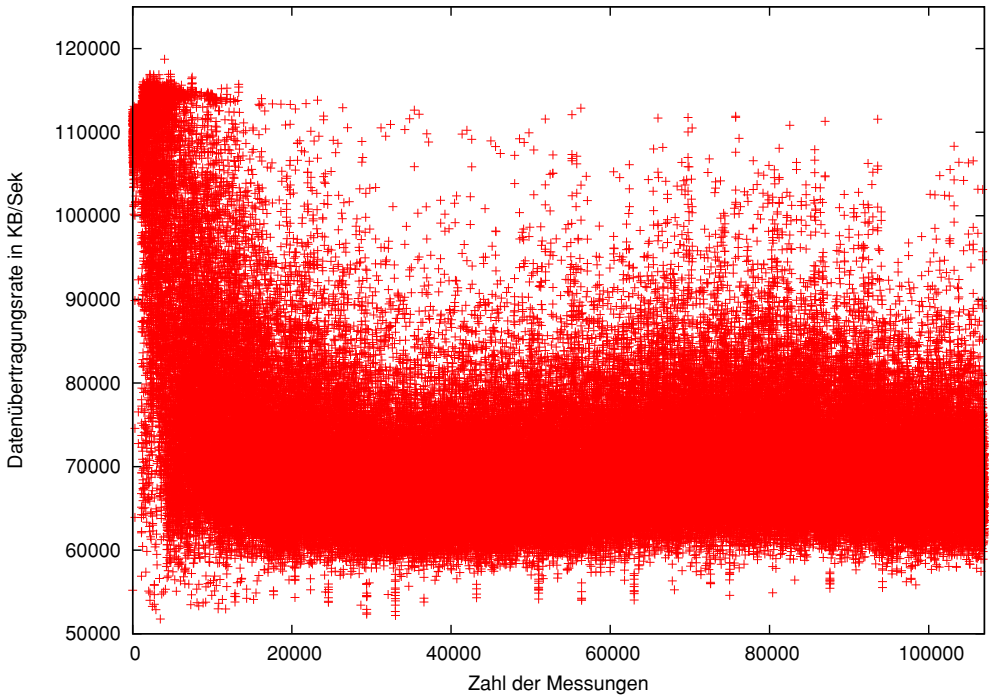


Abbildung 5.9: Auslastung Netzwerk ohne Logging (*Conky*)

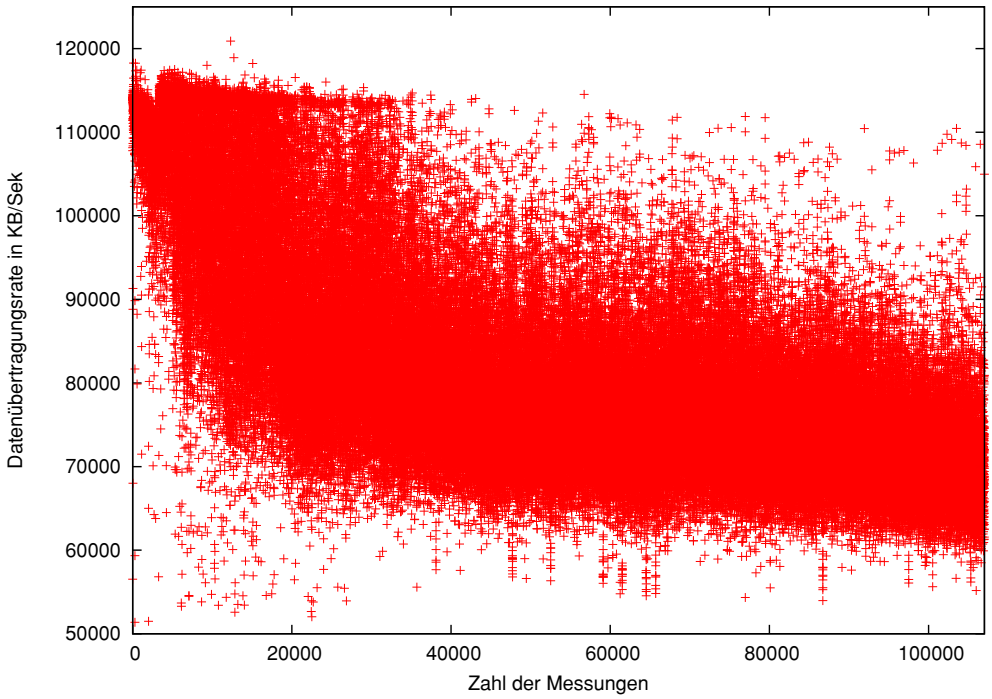


Abbildung 5.10: Auslastung Netzwerk mit Logging (*Conky*)

# Kapitel 6

# Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Framework erstellt, das die Erzeugung neuer Testdatensätze für IDS ermöglicht.

Zu Beginn der Arbeit wurden Anforderungen ausgearbeitet, die von Datensätzen erfüllt werden sollten, um aussagekräftige Ergebnisse zu liefern. Dies sind: eine gute Dokumentation, realistische Annahmen bei der Auswahl der Testumgebung als auch von Angriffen und Hintergrundverkehr, außerdem Interaktion zwischen Angriffs- und Hintergrundverkehr. Daraufhin wurden drei populäre vorhandene Datensätze vorgestellt. Diese wurden dahingehend untersucht, ob sie die zuvor formulierten Anforderungen erfüllen. Die Ergebnisse dieser Auswertung zeigen einige gravierende Schwächen auf. So basieren alle Datensätze auf einem Ursprünglichen, dem Darpa 98 Datensatz. Dieser lässt für sich allein schon mehrere Schwächen erkennen, die auch in den folgenden Datensätzen nicht behoben werden. Deutliche Mängel traten bei der Dokumentation des Datensatzes auf. Aber auch die realistischen Annahmen der Testumgebung sind nicht mehr für den Einsatz mit aktuellen IDS geeignet. So sind das modellierte Rechnernetz sehr klein und die Datenraten zu gering. Sehr kritisch ist auch der Einsatz eines Traffic-Generators zur Erzeugung von Hintergrundverkehr. Durch diese Maßnahme ist keine Interaktion von Angriffs- und Hintergrundverkehr möglich. Da Datensätze jedoch ihren Nutzen für die Entwicklung von IDS bewiesen haben, sollen Neue erzeugt werden.

Im Kapitel 3 wurde ein Ansatz vorgestellt, um neue Datensätze zu erzeugen. Mithilfe eines Frameworks soll eine Vielzahl neuer Datensätze aufgezeichnet werden können. Zunächst wurde festgehalten, welche Funktionalitäten das Framework bieten soll. Um Flexibilität und Erweiterbarkeit zu ermöglichen, wurde das Framework in mehrere Module aufgeteilt. Kernmodule des Frameworks sind das Logging sowie Angriffs- und Hintergrundverkehrserzeugung. Diese sollen unabhängig voneinander agieren und durch ein Steuerungsmodul verwaltet werden. Das Steuerungsmodul soll die Datensatz-Erzeugung zudem automatisiert durchführen können, so dass vom Benutzer lediglich Parameter zu Beginn der Aufzeichnung eingegeben werden müssen.

Das Framework wurde in Kapitel 4 mithilfe von Shell-Skripten implementiert. Für das

Logging der Netzwerkpakete wurde ein auf *iptables* basierender Ansatz gewählt. Es wird das „owner matching“ in Verbindung mit *nflog* sowie dem Logging-Programm *ulogd2* genutzt. Weitere Ansätze schieden wegen ihrer Softwareanforderungen oder dem nicht lückenlosen Logging aus. Die Implementierung jeder Komponente gliedert sich in drei Schritte: die Vorbereitung, Durchführung und Nachbearbeitung. Zudem sind jeweils ein Beispielangriff und ein Typ Hintergrundverkehr implementiert worden. Die Schlüsselerzeugung, die im Rahmen der Nachbearbeitung der Aufzeichnung erfolgt, wurde in Java implementiert.

Nach der Implementierung des Frameworks wurde dieses bewertet. Analog zu den vorhandenen Datensätzen wurde dies anhand der Anforderungen aus 2.2 durchgeführt. Ergebnisse dieser Auswertung sind, dass alle Anforderungen erfüllt werden. Da dem Logging eine zentrale Rolle innerhalb des Frameworks zukommt, wurde zusätzlich eine Performanz-Analyse der Logging-Komponente durchgeführt. Dazu wurden Werte mit und ohne Ausführung des Loggings ermittelt und miteinander verglichen. Die Ergebnisse dieser Untersuchungen zeigen, dass der Ressourcenverbrauch bei Prozessor und Hauptspeicher in einem geringen Maß bleiben und die Ausführung weiterer Prozesse nicht eingeschränkt wird. Überraschend ist, dass die Netzwerk-Übertragungsrate mit eingeschaltetem Logging steigt.

Zusammenfassend lässt sich das Framework als innovativer Ansatz bezeichnen, mit dem personalisierte Datensätze erzeugt werden können, die zudem das gezielte Testen anomaliebasierter IDS zulassen. Darüber hinaus können neue Angriffe jederzeit in die Auswahl der Angriffskomponente aufgenommen werden.

Da auch zukünftig Datensätze benötigt werden, sollte das vorgestellte Framework weiterentwickelt werden. In weiterführenden Arbeiten sollten die Angriffs- und Hintergrundverkehrskomponente weiter ausgebaut werden. Dieses ist durch das erweiterbare Design der Module problemlos mit den vorhanden Teilen möglich. Für die Erweiterung der Angriffskomponente eignen sich zum Beispiel Angriffe aus 2.1. Darüber hinaus soll die Hintergrundverkehrskomponente dahingehend erweitert werden, dass neben der Angabe des Typs von Hintergrundverkehr auch ein Mengenschema als Parameter übergeben werden kann. Mithilfe dieses Mengenschemas sollen Nutzungscharakteristiken parametrisiert werden. Es soll in Form einer mathematischen Funktion übergeben werden und die Menge des erzeugten Hintergrundverkehrs steuern. Mögliche Szenarien sind konstanter, ansteigender oder abnehmender Netzwerkverkehr. Unter Verwendung komplexerer Funktionen könnte beispielsweise eine Tageszeit bedingte Nutzungscharakteristik modelliert werden. Für ein Unternehmensnetz wäre zum Beispiel eine Charakteristik typisch, bei der tagsüber viel Verkehr erzeugt wird, nachts hingegen wenig. Durch das Auftreten anormaler Aktivitäten, wie der Erzeugung von Netzwerkverkehr in der Nacht, kann die Verhaltenserkennung von anomaliebasierten IDS überprüft werden. Im Rahmen weiterer Arbeiten sollte zudem die Steuerungskomponente durch das in 4.3 vorgestellte GPLMT umgesetzt werden, das eine Vielzahl nützlicher Funktionen für das Framework bietet.

Mit den fertiggestellten Komponenten sollte das Framework dann in einer ausreichend großen Testumgebung eingesetzt werden, welche die Anforderungen an Topologie und Datenrate erfüllt.

# Literaturverzeichnis

[1] "Security threat report 2014," Sophos Ltd., Tech. Rep., 2013.

[2] B. für Sicherheit in der Informationstechnik, "Fokus it-sicherheit 2013," Godesberger Allee 185 - 189, 53175 Bonn, July 2013.

[3] B. Städter. (2006, November) http://www.computerlexikon.com/begriff-ddos-attacke.

[4] "Industrial control system security - top 10 bedrohungen und gegenmaßnahmen," Bundesamt für Sicherheit in der Informationstechnik, Tech. Rep., 2014.

[5] I. Shadrin, "It security risks survey 2014: A business approach to managing data security threats," Kaspersky Lab, Tech. Rep., 2014.

[6] A. D. Crenshaw, "Plug and prey: Malicious usb devices," http://www.irongeek.com/downloads/Malicious

[7] J. Allar. (2012, September) http://www.kb.cert.org/vuls/id/649219.

[8] D. G. Nikos Virvilis, "The big four - what we did wrong in advanced persistent threat detection?" in *2013 International Conference on Availability, Reliability and Security*, 2013.

[9] *The 1998 Intrusion Detection Off-line Evaluation Plan*, http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/files/id98-eval-ll.txt, Lexington, MA, 1998.

[10] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, 2000. [Online]. Available: http://doi.acm.org/10.1145/382912.382923

[11] L. L. M. I. of Technology. (1998, March) Simulation network hosts - 1998. [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/docs/hosts_1998.html

[12] K. Kendall, “A database of computer attacks for the evaluation of intrusion detection systems,” Ph.D. dissertation, Massachusetts Institute of Technology, June 1999.

[13] L. L. M. I. of Technology. (1998, March) Documentation. http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/docs/index.html.

[14] G. Turner. (2011, March) http://www.iana.org/assignments/media-types/application/vnd.tcpdump.pcap.

[15] T. U. of Waikato. Arff (book version). http://weka.wikispaces.com/ARFF+%28book+version%29.

[16] D. J. Weber, “A taxonomy of computer intrusions,” Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, ser. CISDA’09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 53–58. [Online]. Available: http://dl.acm.org/citation.cfm?id=1736481.1736489

[18] *Usefulness of DARPA dataset for intrusion detection system evaluation*, vol. 6973, 2008. [Online]. Available: http://dx.doi.org/10.1117/12.777341

[19] C. Elkan. (1999, September) Results of the kdd’99 classifier learning contest. http://cseweb.ucsd.edu/ elkan/clresults.html.

[20] M. Tavallaee, “An adaptive hybrid intrusion detection system,” Ph.D. dissertation, THE UNIVERSITY OF NEW BRUNSWICK, October 2011.

[21] C. Elkan. (1999) Kdd cup 1999: Computer network intrusion detection. http://www.kdd.org/kdd-cup-1999-computer-network-intrusion-detection.

[22] A. S. Adetunmbi A.Olusola. and D. O.Abosede, “Analysis of kdd ’99 intrusion detection dataset for selection of relevance features,” in *Proceedings of the World Congress on Engineering and Computer Science 2010 Vol I*, October 2010.

[23] J. Corbet. (2007, October) http://lwn.net/Articles/256389/.

[24] M. Berg. (2000, May) Zugegriffen am: 11.01.2015.

[25] D. Borkmann. (2014, January) https://git.netfilter.org/iptables/commit/?id=6465867eb48506687872b838b1dd

[26] P. N. Ayuso. (2014) http://netfilter.org/projects/libnftnl/. Zugegriffen am: 11.01.2015.

[27] C. Hellwig. (2005, August) http://mirror.linux.org.au/linux/kernel/v2.6/ChangeLog-2.6.14.

[28] van Hauser. (2014, December) https://www.thc.org/thc-hydra/.

[29] O. T. Matthias Wachs, *Using GPLMT for Large-Scale Deployment and Experimentation - User Guide*, https://gnunet.org/svn/eclectic/gplmt/docs/gplmt-userguide.pdf, München, BY, 2013.

[30] G. Giacobbi. (2006, November) http://netcat.sourceforge.net/. Zugegriffen am: 11.01.2015.