Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Meik Jejkal

Evaluation und Analyse verschiedener Werkzeuge zur Sammlung sowie Auswertung von passiven DNS-Daten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Meik Jejkal

Evaluation und Analyse verschiedener Werkzeuge zur Sammlung sowie Auswertung von passiven DNS-Daten

Abschlussarbeit
eingereicht im Studiengang technische Informatik (BS)
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Klaus-Peter Kossakowski
Zweitgutachter : Martin Hübner

Abgegeben am 20.7.2015

Meik Jejkal

Thema der Arbeit
Evaluation und Analyse verschiedener Werkzeuge zur Sammlung sowie Auswertung von passiven DNS-Daten.

Stichworte
passiveDNS, DNS, Netzwerkanalyse, Flussdaten, IT-Sicherheit, Softwarevergleich, Erfahrungsbericht

Kurzzusammenfassung
EDV-Systeme haben mittlerweile Einzug in viele Bereiche unseres Lebens gehalten und verarbeiten sensible Daten. Besonders der Netzwerkbereich bietet eine Vielzahl von Angriffsvektoren. Ziel dieser Arbeit ist es die Bedeutung von passiveDNS im Bereich der IT-Sicherheit aufzuzeigen und die verfügbaren Softwareprodukte methodisch miteinander zu vergleichen. Stärken und Schwächen der Softwareprodukte werden aufgezeigt und diskutiert.

Meik Jejkal

Title of the paper
Evaluation and analysis of various tools for collection and analysis of passive DNS data

Keywords
passiveDNS, DNS, Network Analysis, Flowdata, IT security, software Comparison, review

Abstract
IT systems are an important part of our daily life and often take care of sensitive data. Especially networks provide a variety of attack vectors. This paper shows the importance of passiveDNS in IT security and compares the available software products. Strengths and weaknesses of those software products will be demonstrated and discussed.

Inhaltsverzeichnis

1 Einleitung 6

1.1 Ziel dieser Arbeit 7

1.2 Zielgruppe dieser Arbeit 7

1.3 Voraussetzungen und Abgrenzungen 7

1.4 Struktur der Arbeit 8

1.5 Definitionen und Konventionen 8

2 DNS – Domain Name System 11

2.1 Übersicht des DNS 11

2.2 Angriffsformen auf DNS 14

2.2.1 DNS-Flooding 14

2.2.2 DNS-Spoofing 15

2.2.3 DNS Amplification Attack 16

2.2.4 DNS-Changer 17

2.3 Was ist passive DNS? 18

2.4 Fallbeispiele 19

2.5 Passive DNS Datenbanken 21

3 Untersuchung verfügbarer passiver DNS Werkzeuge 22

3.1 Aktuell verfügbare passive DNS Werkzeuge 22

3.1.1 PassiveDNS 22

3.1.2 Tshark 24

3.1.3 YAF 25

3.1.4 Farsight DNS Sensor 26

3.1.5 DNSCAP 27

3.1.6 passiveDNS-client 28

3.2 Entwicklung einer Bewertungsmatrix für passive DNS Werkzeuge 30

3.3 Werkzeuganalyse 31

3.3.1 Dokumentation 31
3.3.2 Support 33
3.3.3 Performance 34
3.4 Stärken 42
3.5 Schwächen 42
3.6 Anwendung der Bewertungsmatrix 43
3.7 Ergebnisse 44
4 Erfahrungsbericht zu YAF 45
4.1 Einrichtung 45
4.2 Verwendung mit super_mediator 46
4.3 Export in MySQL Datenbank 49
5 Fazit 53
5.1 Was wurde erarbeitet? 53
5.2 Wurde das Ziel erreicht? 53
5.3 Ausblick 54
Literaturverzeichnis 55
Darstellungsverzeichnis 59
Anhang 61

1 Einleitung

Das Gebiet der IT-Sicherheit ist sehr facettenreich. Zahlreiche Fälle in der Vergangenheit zeigen, dass es immer mehr darauf ankommt diesem Thema Beachtung zu schenken. Als prominente Beispiele seien an dieser Stelle *CVE-2014-0160* [19], besser bekannt als Heartbleed, oder *CVE-2014-3566 [20],* genannt *Poodle*. Diese Sicherheitslücken gefährdeten das weit verbreitete SSL Protokoll zur Datenverschlüsselung. *Heartbleed* machte es möglich, durch gezielte Anfragen, Speicher des angegriffenen Systems auszulesen. Benutzernamen, Kennwörter, gesicherte Sitzungen, oder andere vertrauliche Informationen können im Speicher abgelegt sein. *Poodle* provoziert einen *„Fall Back“* auf das alte SSL 3.0 Protokoll, welches schon seit geraumer Zeit als unsicher gilt. Es bietet eine Vielzahl von Angriffsmöglichkeiten. Viel interessanter sind allerdings Sicherheitslücken, die nicht in den Medien thematisiert werden. *Zero-Day Exploits*, Sicherheitslücken die noch nicht bekannt sind, werden im *Darknet* als Ware verkauft. Dies bietet völlig neue Möglichkeiten, Systeme mit Schadsoftware zu infizieren.

EDV-Systeme haben mittlerweile Einzug in viele Bereiche unseres Lebens gehalten und verarbeiten sensible Daten. Hierunter fallen beispielsweise auch Firmengeheimnisse, deren Verlust mitunter ganze Existenzen gefährden können. Große Teile des Umsatzes werden immer selbstverständlicher über Netzwerke wie das Internet generiert. Werden diese Infrastrukturen mit bösartigem Hintergrund aus verschiedensten Motiven heraus angegriffen, hat dies oft negative Folgen. Hierbei ist auch das Image des Betroffenen in Gefahr, wenn es beispielsweise um den Schutz sensibler Daten geht. Öffentliches Vertrauen lässt sich sehr schwer wieder herstellen, wenn es einmal verloren wurde. Die Größe eines Unternehmens oder seine wirtschaftliche Stellung spielt dabei keine Rolle, jeder kann plötzlich zur Zielscheibe werden. Mitarbeiter werden durch zahlreiche Warnmeldungen desensibilisiert und handeln entsprechend unvorsichtig. Unbekannte E-Mail Anhänge zu öffnen gehört hier genauso dazu, wie beliebige Speichermedien ohne vorherige Überprüfung zu verwenden. In vielen Fällen wird blind auf die Sicherheit der eigenen IT-Infrastruktur vertraut. Dies sind nur einige Aspekte um günstige Bedingungen für einen Angriff auf ein EDV-System zu schaffen.

Um eine Analyse derartiger Vorfälle zu vereinfachen, gibt es auf der Netzwerkebene das Konzept der *passiveDNS*. Dabei wird an einer beliebigen Stelle im Netzwerk der Datenverkehr passiv durch einen Sensor aufgezeichnet. Passiv bedeutet, dass der Datenverkehr nicht beeinflusst, sondern nur mitgeschnitten wird. Tritt nun ein Sicherheitsvorfall auf, so kann zwecks Aufklärung auf die gespeicherten Daten des entsprechenden Zeitraumes zurückgegriffen werden.

1.1 Ziel dieser Arbeit

Ziel dieser Arbeit ist es die Bedeutung von passiveDNS im Bereich der IT-Sicherheit aufzuzeigen und die verfügbaren Softwareprodukte methodisch miteinander zu vergleichen. Hierzu werden die erforderlichen Bewertungskriterien aufgestellt. Diese umfassen sowohl die Dokumentation als auch die Performance und den Support der jeweiligen Produkte. Um besonders wichtige Aspekte herausstellen zu können, werden die DNS Werkzeuge einem Bewertungssystem unterworfen, dessen Ergebnisse zum besseren Verständnis grafisch darstellt werden. Stärken und Schwächen der Softwareprodukte werden aufgezeigt und diskutiert.

Mit den Ergebnissen dieser Arbeit ist es möglich, eine optimale Auswahl von Werkzeugen zu treffen, um passiveDNS Daten sammeln und auswerten zu können. Anhand von Beispielen wird gezeigt, wie sich passiveDNS sowohl präventiv als auch forensisch in verschiedenen Szenarien einsetzen lassen. Die Gefahren, die durch gezielte DNS Angriffe bestehen, werden ebenfalls thematisiert.

1.2 Zielgruppe dieser Arbeit

Diese Arbeit richtet sich primär an Personen, die im Bereich IT-Sicherheit tätig sind, oder deren Interessensgebiet dort liegt. Hierzu zählen auch System- oder Netzwerkadministratoren, die sich im Zuge ihrer Tätigkeit zu einem gewissen Teil mit IT-Sicherheit beschäftigen und Werkzeuge suchen, um DNS Netzwerkverkehr überwachen und auswerten zu können. Um auch fachfremden Personen den Zugang zu der vorliegenden Arbeit zu ermöglichen werden die notwendigen Grundlagen dargestellt. Dazu gehört auch ein ausführliches Glossar mit Begriffserläuterungen.

1.3 Voraussetzungen und Abgrenzungen

Da DNS ein umfangreiches Themengebiet ist, beschränkt sich diese Arbeit auf das benötige Grundwissen, um passiveDNS verstehen und einordnen zu können. Es werden zudem keinerlei Details zu Bewertungsalgorithmen von bestehenden DNS-Datensätzen behandelt, oder mathematische Analysen durchgeführt. Die Auswahl der Werkzeuge ist selbstverständlich auch keinesfalls vollständig, sondern bildet lediglich einen möglichst repräsentativen Teil des momentan aktuellen Angebots ab.

1.4 Struktur der Arbeit

Kapitel 2 behandelt das nötige Vorwissen über DNS und stellt Angriffsszenarien vor. Um eine gemeinsame Grundlage zu schaffen wird im Detail auf passiveDNS und deren Bedeutung eingegangen. Das Ende des Kapitels bilden Fallbeispiele von Angriffen, bei denen passiveDNS maßgeblich zur Aufklärung geführt haben. In Kapitel 3 werden die zur Untersuchung ausgewählten Werkzeuge vorgestellt. Kriterien zur Bewertung der verschiedenen Werkzeuge werden entwickelt und diese anschließend mit einem gewichtetem Bewertungssystem in einer Matrix visualisiert. Nachdem die ausgewählten Werkzeuge vorgestellt und analysiert wurden, folgt eine Diskussion der jeweiligen Stärken und Schwächen. Am Ende von Kapitel 3 werden die Ergebnisse der Untersuchungen vorgestellt. In Kapitel 4 wird das DNS- Werkzeug mit den meisten positiven Bewertungen beschrieben. Das abschließende Kapitel 5 beinhaltet ein Resümee der vorliegenden Arbeit, sowie einen Ausblick auf weitere Aspekte, die für nachfolgende Untersuchungen in diesem Bereich interessant sein könnten.

1.5 Definitionen und Konventionen

Anycast: Ein *„Loadbalancer“* der verteilte Systeme nach außen hin unter einer IP-Adresse ansprechbar macht. Dabei wird versucht, immer die kürzeste Route zum nächsten Server zu verwenden.

AutoWerkzeug: Eine Sammlung von Werkzeugen um Softwarepakete auf verschiedenen Unix Systemen kompilieren und installieren zu können.

AV-Software: Anti Virus Software soll das Herunterladen und Ausführen von Schadsoftware verhindern.

Botnetz: Ein Netzwerk aus zuvor durch Schadsoftware infizierten Systemen, dass von entfernten Angreifern ferngesteuert werden kann.

CVE (*Common Vulnerabilities and Exposures):* Ein Industriestandard zur Beschreibung von Sicherheitslücken.

Darknet: Gesonderter Bereich des Internets, auf den nur über Proxy Server zugegriffen werden kann.

Deep Packet Inspection: Detaillierte Analyse von Netzwerkpaketen auf Protokolle oder weitere Metadaten.

Denial-of-Service: Ein Zustand, in dem ein informationstechnisches System nicht mehr in der Lage ist, Funktionen zu erfüllen für die es vorgesehen wurde.

DLL: hier: dynamische Bibliotheken innerhalb von Windows Betriebssystemen.

DoS: Ein Zustand in dem das System seinen Dienst nicht mehr zur Verfügung stellen kann.

GUI (Graphical User Interface): Eine grafische Oberfläche zum Steuern von Software.
Handshake: Ein Sicherheitsschritt bei dem geheime Informationen ausgetauscht werden um die Identität der beteiligten Parteien sicherzustellen.

HTTP: Ein Transportprotokoll das auf TCP aufsetzt.

IRC: Ein Chatprotokoll mit Funktionalität zum Datenaustausch.

Keylogger: Software oder Hardware die Tastatureingaben protokolliert.

Loopback Interface: Das *Loopback* Interface zeigt auf das eigene System. Es hat üblicherweise die Adresse 127.0.0.1 oder auch *localhost*.

Makefile: Eine Datei die Informationen zum kompilieren und installieren von Softwarepaketen enthält.

Man-in-the-middle-attack: Eine Angriffsform, bei der der Angreifer sich in die Kommunikation einschaltet und so gesendete sowie empfange Daten abhören und manipulieren kann.

MySQL: Ein relationales Datenbanksystem zur Speicherung und Verwaltung von Daten.

Phyton: Eine Skriptsprache mit verschiedenen Programmierparadigmen.

Polling: Aktives Warten auf bestimmte Prozesse mit stetigen Anfragen.

Readme: Dokumentationsdatei in der üblicherweise Installationshinweise, sowie eine kurze Benutzeranleitung zu finden sind.

Schadsoftware: Programmcode, der in seiner Hauptsache dazu dient, schadhafte Handlungen am System durchzuführen. Hierzu zählen unter anderem das Löschen und

Manipulieren von Daten, das Bereitstellen von Fernzugriffen, oder das Aufzeichnen von Eingabegeräten.

Schutzziele in der IT Sicherheit:

Vertraulichkeit: Daten dürfen ausschließlich von autorisierten Benutzern gelesen bzw. modifiziert werden.

Integrität: Daten dürfen nicht unbemerkt manipuliert werden. Änderungen müssen protokolliert werden.

Verfügbarkeit: Die Daten müssen, innerhalb eines gesteckten Zeitrahmens, stets verfügbar sein. Systemausfälle müssen möglichst verhindert werden.

Signatur: Ein elektronischer Fingerabdruck, der die Herkunft von Daten sicherstellen kann. Häufig realisiert durch das Public-Key-Verschlüsselungsverfahren. Dabei wird der Hashwertes der Daten, mit dem privaten Schlüssel des Absenders verschlüsselt.

Social Engineering: Beeinflussung von Personen durch soziale Aspekte. Das persönliche Umfeld wird analysiert, um Identitäten vorzutäuschen, oder bestimmte Verhaltensweisen anzuwenden. Das Opfer soll so zu einem gewollten Verhalten gebracht werden.

TCP/UDP: Protokolle des OSI Layer 4 zur Abwicklung von Datenverkehr.

TTL: Time-To-Live beschreibt die Zeit, nachdem ein DNS Eintrag „abgelaufen“ ist und erneut vom primären DNS-Server der entsprechenden Zone abgerufen werden muss.

Unix Zeitstempel: Eine Zeitdefinition bei der die Sekunden seit dem 01.01.1970 00:00 Uhr gezählt werden.

Versionsverwaltung (GIT): Eine Werkzeug mit dem sich Programmcode von unterschiedlichen Nutzern verwalten lässt.

Whois: Eine Anfrage an einen Domain Registrator nach den Personendaten einer Domain.

2 DNS – Domain Name System

2.1 Übersicht des DNS

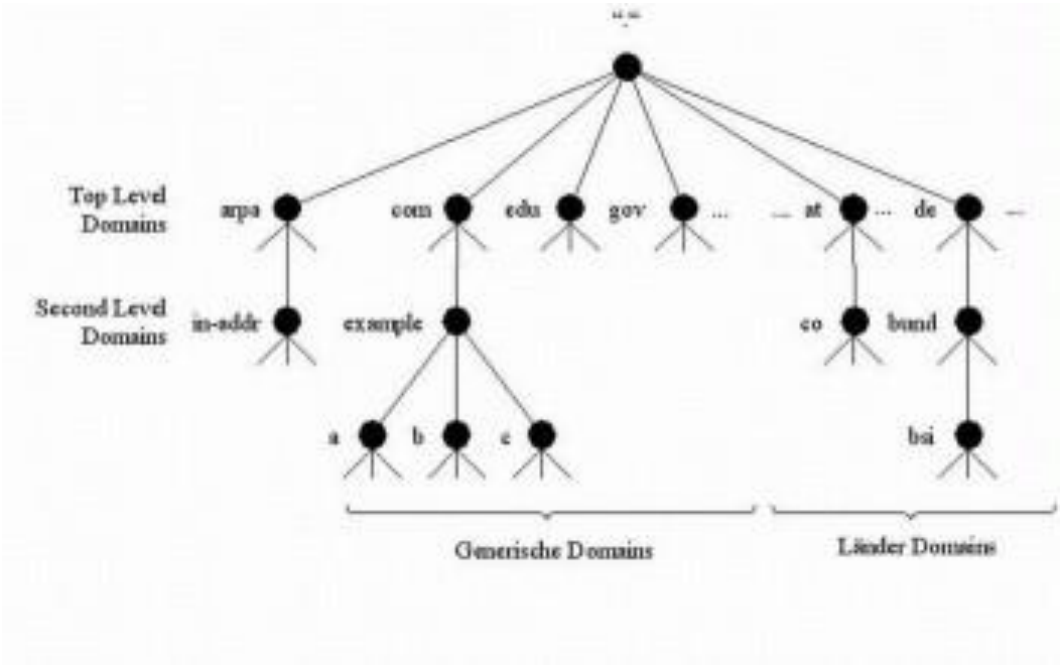
Das DNS (Domain Name System) ist ein global verteilter Verzeichnisdienst, der den Namensraum des Internets verwaltet. Der Dienst ist baumartig mit Blättern und Knoten strukturiert und unterliegt einer Hierarchie [2]. Es wurde 1983 von Paul Mockapetris entworfen und ist u.a. in den RFC Standards 882 sowie 883 festgelegt. Diese wurden später von RFC1034 und RFC1035 ersetzt [24]. DNS übersetzt Domain Namen in IP-Adressen und umgekehrt. Dies vereinfacht die Nutzung von Netzwerken und des Internets. IP-Adressen bestehen im Falle von IPv4 aus 4 Blöcken, die durch Punkte getrennt werden und Werte von 0-255 annehmen können (Beispieladresse: 255.255.255.255). Im Falle von IPv6 handelt es sich um 128 Bit große Adressen, die zur Vereinfachung hexadezimal notiert werden (Beispieladresse: 2001:db8:85a3::8a2e:370:7344) [10].
Um einen entfernten Dienst nutzen zu können, wird die Adresse des bereitstellenden Servers benötigt. Da IP-Adressen selten im natürlichen Sprachgebrauch verwendet werden, fällt es schwer sich diese zu merken. Wörter und Bezeichnungen sind ein gängiges Mittel um sich Sachverhalte einzuprägen. Es wird also ein Verzeichnis benötigt, in dem IP-Adressen mit Bezeichnungen verknüpft werden. Eine weit verbreitete Analogie ist die des Telefonbuches, in dem die physikalischen Adressen zu bestehenden Namen zu finden sind. Dabei werden die DNS Einträge als *Resource Records* (RR) bezeichnet. Diese können von verschiedenem Typ sein [18].

Resource Record Typ	***Erläuterung***
SOA	Grundeintrag einer Zone. Enthält Daten über Besitzer, autoritären Server, verantwortliche Person, Seriennummer, Aktualisierungs- und Ablaufzeitzeit, sowie *TTL*
NS	Nameserver
A	Enthält die IP Adresse des Servers für die Vorwärtsauflösung
PTR	Enthält den Domainnamen für die Rückwärtsauflösung
CNAME	Alias für anderen Domainnamen
MX	IP-Adresse des Mail Servers
SRV	Enthält IP-Adressen für weitere Dienste

[Dar. 1 DNS Resource Record Typen] [18]

„Resource Records“ werden in Zonendateien eingetragen und verwaltet. Die Zonendaten der autoritären Server werden, gemäß der Hierarchie, regelmäßig von der darunter liegenden Ebene angefragt. Den Zeitpunkt hierfür legt der *SOA Resource Record* fest. Die Seriennummer dieses Eintrags dient dazu, die Version der Zonendatei auf mehreren Servern der gleichen Zone konsistent zu halten. Wird bei einem Zonenabgleich eine höhere Seriennummer festgestellt, so wird diese Zonendatei übernommen [18].

Die Namensauflösung findet, wie oben bereits beschrieben, hierarchisch in Form eines Baumes statt.



[Dar. 2 Hierarchischer DNS Baum BSI] [2]

Dar. 2 zeigt den schematischen Aufbau der DNS Hierarchie. Die höchste Ebene, die Baumwurzel, wird als Punkt dargestellt und wird *root* genannt. Auf der folgenden Ebene liegen die *Top-Level-Domains*, wie beispielsweise *de*, *com*, *at*. Darauf folgen die *Second-Level-Domains* wie *co*, oder *bund*. Soll ein Domainname aufgelöst werden, so versucht das anfragende System zunächst die zugehörige IP-Adresse im eigenen DNS Cache zu ermitteln. Ist dies erfolglos, wird die systemeigenen Hosts Datei zur Namensauflösung herangezogen. Ist hier kein Eintrag für den entsprechenden Domainnamen zu finden, wird die Anfrage über den systemeigenen Resolver DNS konform umgewandelt und an den bevorzugten DNS-Server geschickt. Dieser prüft nun seinen Cache auf einen entsprechenden Resource Record. Ist kein passender Eintrag vorhanden, so wird die Anfrage an einen der DNS Root Server weitergeleitet, der die IP-Adresse des zuständigen Nameservers zurückliefert. Hier kann nun die gesuchte IP-Adresse des aufzulösenden Domainnamens angefragt werden.

Etwaige Subdomains werden von diesem Server ebenfalls aufgelöst. Wurde der angefragte Domainname erfolgreich aufgelöst, so wird die entsprechende IP-Adresse für eine gewisse Zeit (TTL) im lokalen Cache des anfragenden Systems zur weiteren Verwendung gespeichert [2] [24].

Es existieren 13 DNS Root Server [25]. Physikalisch handelt es sich dabei aber um mehrere hundert Server verteilt über die ganze Welt. Mittels Anycast werden sie redundant betrieben. Diese Adressierungsart sorgt dafür, dass immer derjenige Server angesprochen wird, der vom anfragenden System aus die kürzeste Route hat.

Um die Sicherheit von DNS zu erhöhen, wurden eine Reihe von Internetstandards unter dem Namen Domain Name System Security Extensions (DNSSEC) definiert. Diese sollen die Authentizität sowie Integrität der Ressource Records sicherstellen. Die Vertraulichkeit wird hierbei außen vor gelassen, eine Verschlüsselung ist nicht vorgesehen. Im Kern wird bei DNSSEC der Resource Record zusätzlich um eine digitale Signatur erweitert. Der Resolver kann nun, mittels des öffentlichen Schlüssels vom DNS-Server, den Resource Record verifizieren. Angreifern wird es so erschwert, erfolgreich Attacken, wie DNS Spoofing, durchzuführen, da hierzu auch die digitale Signatur im Resource Record plausibel sein muss [9].

Vor der Einführung globaler DNS-Server waren ausschließlich lokale Dateien die Quelle, aus denen die IP-Adressen zu bestimmten Domainnamen entnommen werden konnten. *„Hosts Dateien“* [26] enthalten den Domainnamen und die entsprechende IP-Adresse in einem einfachen Textformat. Durch die rasant ansteigende Änderungsrate der Domainnamen und IP-Adressen, war es allerdings schnell ein logistisches Problem, die *„Hosts Dateien“* aller Teilnehmer des Netzwerks stets auf einem aktuellen Stand zu halten. Somit löste DNS dieses System kurz nach seiner Einführung 1983 ab [24].
„Hosts Dateien“ werden heute lediglich in lokalen bzw. virtuellen Rechnernetzen oder zu Filterzwecken verwendet. Sie sind aber immer noch vorhanden und dienen, neben dem eigenen DNS Cache, immer noch als erste Quelle der Namensauflösung im System. Durch Angabe des *loopback* Interfaces (127.0.0.1) in der *„Hosts Datei“* lässt sich beispielsweise erreichen, dass bestimmte Anfragen ins Leere laufen, da sie auf das lokale System umgeleitet werden. Eine gängige Praxis bei Schadsoftware ist es, die Domainnamen bekannter Virenhersteller mit dem *loopback* Interface in der „*Hosts Datei*“ zu verknüpfen. Dies soll verhindern, dass die *AV-Software* Kontakt zu ihrem Update Server aufnehmen und ihre Signaturdaten aktualisieren kann. Sind diese veraltet, kann aktuellere Schadsoftware nicht mehr als solche identifiziert werden. Beim Betriebssystem Microsoft Windows ist die *Hosts Datei* beispielsweise versteckt unter dem Pfad *%SystemRoot%\system32\drivers\etc* zu finden. Unter Linux liegt die entsprechende Datei üblicherweise im Verzeichnis *etc/*.

2.2 Angriffsformen auf DNS

Ein Angriff auf DNS kann verschiedene Hintergründe haben. Ein Aspekt sind Denial-of-Service-Angriffe, die das *Schutzziel* Verfügbarkeit betreffen. Dabei wird der DNS-Server [1] durch *DNS-Flooding* in einen DoS Zustand versetzt [7]. DNS-Server können aber auch selbst aktiv genutzt werden um andere Systeme zu beeinträchtigen. Die kann mit Hilfe einer *DNS Amplification Attack* umgesetzt werden [4].
Weiterhin können die Schutzziele Vertraulichkeit und Integrität durch Manipulation von Netzwerkverkehr zwischen DNS Resolver und DNS Server kompromittiert werden. Durch eine Kombination aus DNS-Spoofing und anschließendem *Man-In-The-Middle* Angriff [5] wird es einem Angreifer möglich, an vertrauliche Daten zu gelangen, oder diese manipulieren zu können [6].
Ein weiterer Faktor sind Botnetze. Dabei werden Computer mit Schadsoftware infiziert, die über einen Steuerungsserver Befehle annehmen kann. Dies kann beispielsweise über unauffällige *http* Verbindungen oder *IRC* erfolgen. Die befallenen Systeme werden in einem Netzwerk zusammengefasst. Durch *Fast Fluxing* wird die Adresse des Steuerungsservers verschleiert [14]. Hierzu wird das Round-Robin-DNS System genutzt, welches bei einer DNS Anfrage zur Lastverteilung mit mehreren IP-Adressen antwortet. Aus denen wird eine zufällig gewählt. Die IP-Adressen verweisen nun ihrerseits auf weitere infizierte Computer, die dann erst die Verbindung mit dem Steuerungsserver aufnehmen. Um das System noch undurchsichtiger zu machen, besitzen diese Resource Records eine *TTL* von wenigen Minuten. Dieser schnelle Wechsel von IP-Adressen kann durch eine Analyse von gesammelten DNS-Daten festgestellt werden.

2.2.1 DNS-Flooding

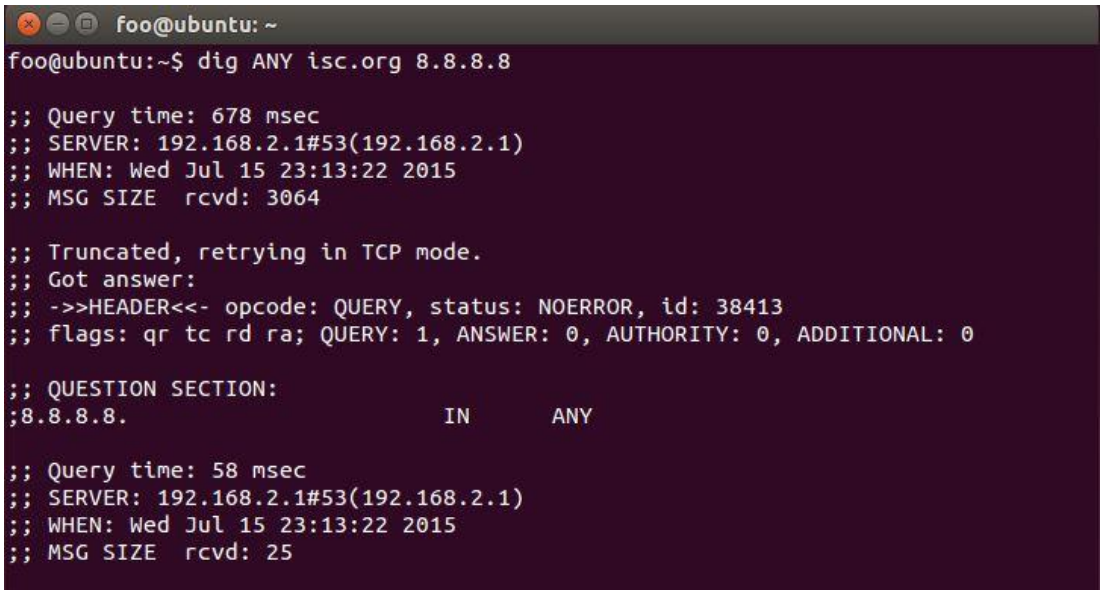
Bei dieser Angriffsart wird der DNS-Server mit einer Vielzahl von Anfragen blockiert. Es stehen so keine Ressourcen mehr zur Verfügung um reguläre Anfragen zu beantworten. Das System befindet sich in einem *Denial-of-Service* Zustand [7].
Es gibt verschiedene Möglichkeiten einen derartigen Angriff auszuführen. Eine Möglichkeit sind DNS Anfragen von Botnetzen, die für die Serverinfrastruktur schwer von normalen Anfragen zu unterscheiden sind. Botnetze lassen sich zudem auch mieten, um sie dann gezielt für derartige Angriffe zu nutzen. Auch ist es möglich direkt Cloud Infrastrukturen zu mieten, um eine große Anfragenlast erzeugen zu können. Diese Umstände vereinfachen es einer breiten Masse, diese Form von Angriffen ohne viel Fachwissen durchführen zu können.

2.2.2 DNS-Spoofing

Spoofing im Allgemeinen steht für die Verschleierung der eigenen Identität meist in Kombination mit der Annahme einer fremden, um sich deren Rechte anzueignen. Im Falle von DNS-Spoofing [3] ist dies die erfolgreiche Zuordnung eines Domainnamens zu einem kontrollierten Server vice versa. Dies kann bei der Auflösung eines Domainnamens durch den *Resolver* des Opfers durchgeführt werden. Ein Angreifer muss hierzu die IP-Adresse innerhalb der DNS Antwort abändern, sodass diese auf ein System unter eigener Kontrolle zeigt. Gelingt das *DNS-Spoofing*, kann der Angreifer an Stelle des regulären DNS-Servers die Anfrage des Opfers mit einer beliebigen IP-Adresse beantworten.
Damit das erste manipulierte Antwortpaket des Angreifers nicht vom Resolver des Opfers verworfen wird, muss dieses vor der regulären Antwort beim Zielsystem ankommen. Um diese notwendige Verzögerung zu erreichen, kann, wie im Kapitel zuvor beschrieben, ein *DoS* Angriff auf den beteiligten DNS-Server durchgeführt werden. Ein lokaler Angreifer, kann Datenpakete über einen falschen Hotspot umleiten, um den gleichen Effekt zu erzielen. Gelingt es dem Angreifer, sich als DNS-Server auszugeben, ist die Wahrung der *Schutzziele* im Sinne der IT-Sicherheit nicht mehr gegeben. Es besteht die Gefahr eines *Man-in-the-Middle* Angriffs [5]. Dabei gibt sich der Angreifer gegenüber dem Sender als Empfänger, und gegenüber dem Empfänger als Sender aus. Die Vertraulichkeit ist kompromittiert, es können sensible Daten wie Kennwörter, PIN‘s, Kreditkarten- oder Kontonummern erlangt werden. Hierzu werden dem Opfer durch Phishing täuschend echte Kopien von Webseiten bekannter Banken oder Firmen untergeschoben, auf denen die entsprechenden Daten eingegeben werden müssen [8].
Wieviel Aufwand es bedeutet, einen solchen Angriff durchzuführen, hängt von der Konfiguration des entsprechenden Netzwerkes ab.
Passive DNS kann in solchen Fällen helfen DNS-Spoofing zu erkennen. Durch einen Abgleich mit einer Datenbank aus Kapitel 2.2.1 können die zurückgegebenen IP-Adressen überprüft werden.

2.2.3 DNS Amplification Attack

Bei diesem Angriffstyp wird ausgenutzt, dass in bestimmten Fällen kurze DNS Anfragen sehr lange Antworten in Form von großen Resource Records erzeugen können.



[Dar. 3 DNS-Anfrage nach isc.org]

Wie in der Darstellung 3 gezeigt, erfolgt auf die 64 Byte Anfrage an einen Google DNS *Resolver* eine 3064 Byte große Antwort. Hier tritt also eine Verstärkung um den Faktor 48 auf [12]. Wird *DNSSEC* auf dem Server eingesetzt, werden die Antworten größer und begünstigen diese Angriffsform zusätzlich. Dies liegt an der digitalen Signatur die den *Resource Records* hinzugefügt wird.
Im Idealfall sollten offene DNS Resolver so konfiguriert sein, dass sie Anfragen nur von autorisierten Teilnehmern beantworten. Im Falle eines firmeninternen DNS Resolvers sollten lediglich Anfragen aus dem eigenen Netzbereich als autorisiert gelten.
In der Praxis wird dies oft nicht beachtet. Aus diesem Grund werden teilweise gefährdete IP-Adressen im Internet veröffentlicht, um die Betreiber darauf aufmerksam zu machen. Eine weitere Möglichkeit große DNS Antworten zu erzeugen, ist der Betrieb eines eigenen DNS-Servers. Damit kann die Größe der entsprechenden Resource Records selbst bestimmt werden.

Eine weitere Vergrößerung des Verstärkungsfaktors lässt sich durch Botnetze erreichen. Diese steigern die Anzahl der Anfragen zusätzlich. Der eigentliche Angreifer tritt dabei nicht

direkt in Erscheinung, sondern steuert das Netz lediglich fern. Anfragen von Botnetzen sind, aufgrund der verschiedenen IP-Adressen der einzelnen Teilnehmer, nur schwer von regulären DNS Anfragen zu unterscheiden.
Eine Analyse gesammelter DNS-Daten kann hier aber dazu dienen, verdächtige DNS Anfragen auszumachen und den betreffenden Netzwerkverkehr durch gezielte Filter wie Firewalls zu blockieren.

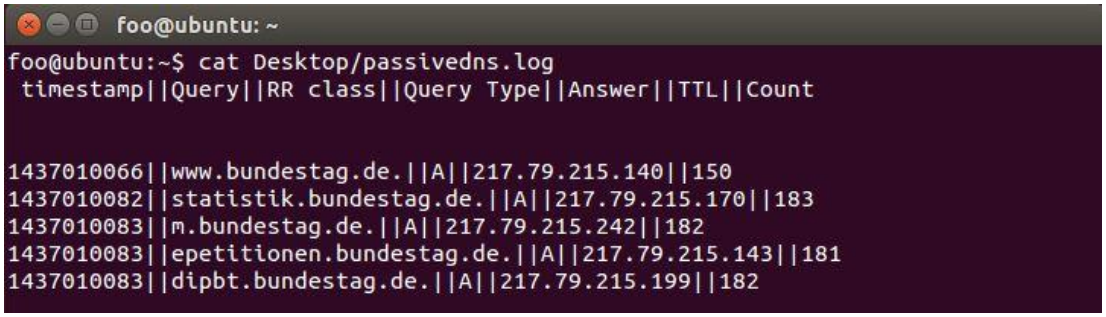
2.2.4 DNS-Changer

DNS-Changer [27] manipulieren den im System eingestellten DNS-Server und leiten die Anfragen an einen DNS-Server um, der sich unter der Kontrolle des Angreifers befindet. Dies kann durch Sicherheitslücken, oder nicht geänderte Standard Zugangsdaten auch bei Routern vorkommen. Dieser Fall ist besonders schwerwiegend, da alle Geräte betroffen sind, die diesen Router als bevorzugten DNS-Server nutzen. Ist der Router gleichzeitig DHCP Server, so wird dieser in der Standardeinstellung als bevorzugter DNS-Server konfiguriert. Diese Konfiguration ist weit verbreitet. *CVE-2015-1187* [13] beschreibt aktuell eine Sicherheitslücke, die es entfernten Angreifern ermöglicht, die Kontrolle über D-Link Router einer bestimmten Serie zu erlangen. Derartige Sicherheitslücken werden zwar häufig zeitnah veröffentlicht, allerdings versäumen es viele Anwender ihre Router Firmware entsprechend zu aktualisieren.
2007 bis 2011 gab es einen berühmten Fall, bei dem die estländische Firma Rove Digital eine Malware Namens *DNS-Changer* in Umlauf brachte, um gezielt DNS Anfragen auf eigene DNS-Server umzuleiten [27]. Hierbei wurden reguläre Werbeeinblendungen durch eigene Werbung ersetzt und so ein Gewinn von geschätzt 14 Millionen US Dollar eingenommen. Betroffen waren sowohl Windows, als auch OS X Unix Systeme. Das FBI übernahm durch eine richterliche Verfügung schließlich die Nameserver, und ließ diese wieder auf die regulären IP-Adressen verweisen. Vor Ablauf der Verfügung, wurden die Server abgeschaltet. Google zeigte auf seiner Startseite an, wenn der Computer von *DNS-Changer* betroffen war und verlinkte zu einer Anleitung, um den Eingriff ins System rückgängig zu machen. Weltweit wurden etwa 4 Millionen infizierte Systeme geschätzt. Betroffene, die den falschen DNS-Server immer noch als primäre Adresse eingestellt hatten, konnten nach der Abschaltung nicht mehr wie gewohnt auf das Internet zugreifen. In diesem Fall wurde nur ein Bruchteil des Potentials dieses Angriffs ausgenutzt. Es wäre auch ohne großen Aufwand möglich gewesen, durch Spoofing über einen langen Zeitraum unbemerkt an vertrauliche Informationen zu gelangen, wie Logins, Kontoinformationen, E-Mail Postfächer etc.. Passive DNS kann in solchen Fällen, durch einen Datenbankabgleich, dazu beitragen derartige Eingriffe ins System aufzudecken, um dagegen vorgehen zu können.

2.3 Was ist passive DNS?

Die Möglichkeiten DNS Informationen zur Klärung von IT Sicherheitsvorfällen zu Rate zu ziehen, sind aufgrund einer fehlenden Historie sehr begrenzt. Es fehlen wichtige Informationen, wie beispielsweise der Zeitpunkt wann ein bestimmter Domainname erstmalig in einem Netzwerk aufgelöst wurde. Auch die DNS Antwort zu diesem Zeitpunkt ist nicht mehr abrufbar.
Es gibt die Möglichkeit mittels *Reverse Lookups* [27] zu einer IP-Adresse zugehörige Domainnamen zu ermitteln. Diese basiert allerdings auf sog. PTR Resource Records in separaten DNS-Zonen. Deren Aktualität und Existenz ist nicht immer gewährleistet, da diese nicht automatisch angepasst werden, wenn sich etwas an der Haupt DNS-Zone verändert. Zudem beschreiben sie nur den Status quo. Informationen darüber, welche Clients einen bestimmten Domainnamen aufgelöst haben, sind ebenfalls nicht verfügbar. Diese können nützlich sein, um im Falle einer Infektion durch Schadsoftware die betroffenen Systeme zu bereinigen.

Vor diesem Hintergrund, entwickelte Florian Weimer ein Konzept für die *Passive DNS Replication* [15] welches 2005 bei der FIRST Konferenz erstmals vorgestellt wurde. Es handelt sich dabei um die Speicherung von DNS relevantem Datenverkehr innerhalb von Netzwerken. Hierbei wird passiv vorgegangen, d.h. der Datenverkehr wird nicht manipuliert und auch nicht über Proxy Server umgeleitet.
Zum Sammeln der Netzwerkdaten können Sensoren beliebig im Netzwerk verteilt werden. Diese können ihre Daten zur weiteren Verarbeitung an eine Hauptinstanz weiterleiten, oder gleich selbst archivieren. Zur Archivierung können Dateien oder auch Datenbanken verwenden werden. Die so entstehende Datensammlung kann durch ergänzende Informationen erweitert werden und bietet eine Vielzahl von Analysemöglichkeiten. Die Folgende Darstellung zeigt einen Auszug aus dem Werkzeug „passiveDNS“, auf das im weiteren Verlauf dieser Arbeit noch näher eingegangen wird.



[Dar. 4 DNS Log PassiveDNS]

Der Zeitstempel ist bei der Analyse sehr nützlich, um den Ursprung verdächtiger Aktivitäten zurückverfolgen zu können. Würde sich die IP-Adresse innerhalb weniger Minuten mehrmals verändern, sollte dies einen Verdacht aufkommen lassen. Weiterhin zu sehen ist die *TTL* und ein Zähler für die Anzahl der Zugriffe auf dieselbe Domain. Durch den Counter müssen gleiche Zeilen nicht wiederholt werden, dies trägt zur Speichereffizienz bei und vereinfacht die Lesbarkeit der Ergebnisse. Gleichzeitig lassen sich viele Einzeldatensätze zu größeren Datenbanken zusammenfassen. Die Daten können zu öffentlichen Anbietern wie Farsight [30] hochgeladen werden.
DNS bietet von sich aus lediglich die Möglichkeit Anfragen nach primären Schlüsseln, Domains, Klassen oder Typen durchzuführen. Dabei wird immer nur der aktuelle Stand abgebildet.
Die Datensätze des Sensors hingegen können durch zusätzliche Informationen weiter ergänzt werden. Hierzu gehören beispielweise *Whois* Anfragen, um auffällig gewordene Inhaberdaten erkennen zu können.
PassiveDNS kann sowohl nach einem IT-Sicherheitsvorfall zur Aufklärung eingesetzt, als auch präventiv als Überwachungsinstanz verwendet werden. Hierbei können die gesammelten Daten permanent auf verdächtige Einträge geprüft und dabei ein Wert gebildet werden. Erreicht dieser Wert festgelegte Grenzen, können automatisiert Aufgaben ausgeführt werden. Diese können vom Verschicken einer E-Mail an den Systemadministrator, bis hin zum direkten Setzen und Aktivieren von Firewall Regeln reichen. Aus Effizienzgründen, sowie zur besseren Verarbeitung der gesammelten Daten, werden diese für gewöhnlich in Datenbanken wie zum Beispiel MySQL gespeichert.

2.4 Fallbeispiele

Die folgenden Fälle zeigen Angriffe, bei denen eine Analyse von DNS-Daten maßgeblich zur Aufklärung geführt hat.

Das erste Beispiel handelt von Google. Anfang 2010 gab der Konzern bekannt, Opfer eines sorgfältig geplanten Angriffes gewesen zu sein, bei dem teils sensible Daten ausgespäht wurden. 20 weitere Firmen sollen ebenfalls davon betroffen gewesen sein, darunter auch Adobe. Der Angriff wurde *Aurora* [21] genannt. Es handelte sich hierbei um Schadsoftware, die *Zero-Day Exploits* im Internet Explorer 6 und dem Adobe Reader ausnutzte, um eine Hintertür zu installieren [22]. Im Anschluss daran, konnte mittels *pass-the-hash* Technik auf den Windows eigenen Passwortspeicher zugegriffen werden. Alternativ konnte auch ein *Keylogger* installiert werden. So wurde es möglich auf interne, sensible Systeme zuzugreifen.
Zuvor wurden gezielt Mitarbeiter ausgewählt, die durch *Social Engineering* auf speziell präparierte Webseiten geleitet wurden, oder E-Mail Anhänge von vermeintlich bekannten Personen geöffnet hatten. Dadurch konnte sich die Schadsoftware in Form einer *DLL* auf dem System installieren. Google verdächtigt die chinesische Regierung, da unter anderem

auch Daten über die Tibetbewegung, oder einzelne kritische Journalisten betroffen waren. Zudem sollte Google seinen Dienst, entsprechend den chinesischen Vorgaben, zensieren, weigerte sich jedoch dies umzusetzen. Der Angriff sollte in einem kleinen Rahmen bleiben, damit der Zugriff auf die Daten möglichst lange erhalten bleibt. Es ging dabei auch um Software Repositories, eine wichtige Ressource von Internetunternehmen.
Innerhalb der Schadsoftware wurde ein fest programmierter Domainname gefunden. Mit Hilfe einer DNS-Analyse auf Basis zuvor gesammelter Daten konnte sowohl das Datum der ersten Infektion als auch die Herkunft der beteiligten Server festgestellt werden. Zudem konnten DNS-Anfragen lokalisiert werden, die zu neu registrierten Webseiten führten, die noch nie aus dem Netzwerk besucht wurden. Ohne eine Sammlung von DNS-Daten, wären derartige Analysen nicht möglich gewesen.
Schadsoftware kann beliebig hoch entwickelt sein. In den meisten Fällen wird aber eine Verbindung nach außen benötigt, um Daten auszulesen oder Kommandos zu vermitteln. An diesem Punkt setzt passive DNS an und liefert eine Basis für tiefergehende Analysen.

Ein weiteres Beispiel ist die Schadsoftware *Conficker [11]*. Dieser nutzt die Sicherheitslücke MS08-067 [17] in Microsoft Windows. Für diese wurde bereits am 23.Oktober 2008 ein Sicherheitspatch veröffentlicht. Monate später waren nach einer Schätzung von *F-Secure* neun Millionen Computer infiziert. Darunter auch die Bundeswehr, oder Behörden in Kärnten. Dieser Fall macht deutlich, wie nachlässig mit der Aktualisierung von Software umgegangen wird. Auf infizierten Systemen verändert die Schadsoftware DNS Schnittstellen. Dies hat zur Folge, dass DNS-Anfragen die bestimmte Stichwörter enthalten ins Leere laufen. Darunter die meisten Namen von AV-Software, sowie Microsoft selbst. Signaturdatenbanken können nicht mehr aktualisiert werden, Windows Updates funktionieren ebenfalls nicht mehr. Zudem versucht sich *Conficker* mit Standardpasswörtern in Kombination mit Benutzerkonten aus der Domänenliste anzumelden. Dies schlägt zwar in den meisten Fällen fehl, allerdings werden Benutzerkonten in der Standardeinstellung nach zehn Fehlversuchen für 30 Minuten gesperrt.
Eine Analyse der DNS-Daten ergab in solchen Fällen, dass Domainnamen die sonst regelmäßig aufgerufen wurden, über ein weit zurückliegendes *lastseen* Datum verfügten. Mit diesem Wissen konnten auch gezielt infizierte Systeme lokalisiert werden, um diese zu bereinigen.

2.5 Passive DNS Datenbanken

Würden passive DNS-Daten nur innerhalb einzelner Organisationen gesammelt und gespeichert, so ließe sich das volle Potential dieser Technik nicht nutzen. Die Verwendung von Netzwerken, beispielsweise im Falle des Internets, variiert je nach Einsatzgebiet sehr stark. Firmen die im Social Media Bereich tätig sind, werden sich in anderen Netzbereichen bewegen, als Unternehmen die sich mit Aktien beschäftigen. Aufgrund der Ausmaße des Internets ist es von Vorteil, DNS-Daten an verschiedenen Stellen zu sammeln und zu konzentrieren, um einen möglichst großen Bereich abdecken zu können.

Um zentrale Datensenken für DNS-Daten zu schaffen, bieten dies einige Anbieter als Service an. Hierzu gehören beispielsweise BFK.de, CIRCL, DNSDB (Farsight), Mnemonic, PassiveDNS.cn, PassiveTotal, RiskIQ, TCPIPUtils sowie VirusTotal. Nach vorheriger Registrierung werden API Keys oder Login Kombinationen bereitgestellt, mit denen man sich bei den Diensten anmelden kann. Im Anschluss daran kann die Datenbank zur Abfrage von Domainnamen oder IP-Adressen genutzt werden. Darüber hinaus kann die jeweilige Datenbank mit Informationen aus eigenen DNS Sensoren gespeist werden. Die Anbieter können untereinander auch beliebig kombiniert werden. Im Laufe der Analyse wird ein Werkzeug vorgestellt mit dem einige dieser Datenbanken abgefragt werden können. Die Ergebnisse können auch visualisiert werden um Netzwerk Topologien leichter erkennen zu können.

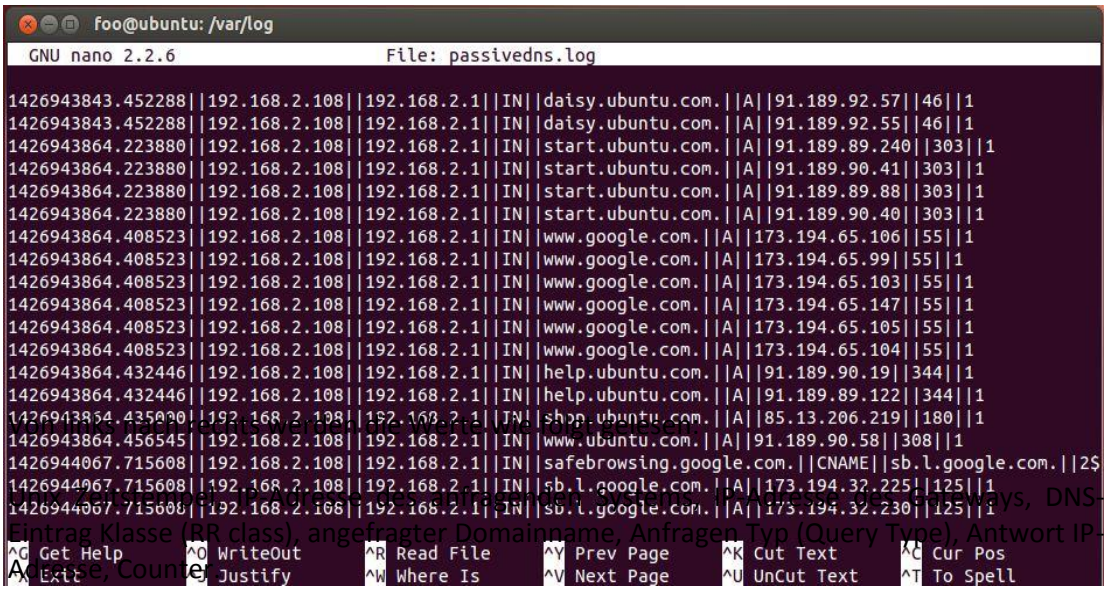
3 Untersuchung verfügbarer passiver DNS Werkzeuge

3.1 Aktuell verfügbare passive DNS Werkzeuge

Im Folgenden wird eine Auswahl verschiedener Werkzeuge zur Verarbeitung von passiven DNS-Daten vorgestellt. Darunter sind Sensoren um Netzwerkflussdaten aufzuzeichnen und um DNS relevante Informationen zu exportieren. Diese haben teilweise auch Zusatzfunktionen und können bereits aufgezeichnete Netzwerkdaten in das *pcap* Dateiformat konvertieren. Auch die Visualisierung von DNS-Daten wird durch ein Werkzeug ermöglicht.

3.1.1 PassiveDNS

Das Werkzeug *PassiveDNS* [28] stammt von Edward Bjarte Fjellskal und wurde in C programmiert. Es bietet die Funktionalität DNS Verkehr passiv, direkt von der Netzwerkschnittstelle aufzuzeichnen, oder schon aufgezeichnete Netzwerkdaten im pcap Format einzulesen. Die DNS relevanten Informationen werden zur weiteren Verarbeitung in ein eigenes Textformat konvertiert und in einer Datei gespeichert.
Die erste Version wurde am 29 April 2011 auf der Plattform GitHub veröffentlicht. Seitdem wurde es stetig weiterentwickelt, bis hin zum letzten Commit am 20. Februar 2015. Der Autor gibt *FreeBSD*, *OS X*, *Ubuntu (10.04 / 12.04)* sowie *RedHat (RHEL 6u2 x86_64*) als kompatible Plattformen an. Zur Installation wurde *autotools* in das Produkt integriert. Die Dokumentation ist einfach gehalten und enthält alle wichtigen Informationen die zur Einrichtung und Bedienung gebraucht werden. Es fehlt lediglich der Hinweis zur Installation zusätzlicher Pakete, die nicht immer fester Bestandteil von Standard Linux Distributionen sind. Hierbei handelt es sich um die Pakete *autoconf*, *build-essential* sowie *libWerkzeug*.
Um die aufgezeichneten Daten komfortabler verarbeiten zu können, enthält das Programmpaket ein Phyton Script, zur direkten Übertragung in eine MySQL Datenbank. Hier können die Daten dann in der gängigen SQL beliebig verknüpft und ausgewertet werden. Eine Datenbank bietet, im Vergleich zu einer regulären Datei, in vielerlei Hinsicht Vorteile. Sie ist nicht nur besser was Geschwindigkeit und Größe anbelangt, innerhalb der Datenbank lassen sich auch zusätzliche Tabellen mit ergänzenden Informationen anlegen.



[Dar. 5 DNS Verkehr der Ubuntu Startseite aufgezeichnet mit Passive DNS]

Dar. 5 zeigt die Ausgabe der extrahierten DNS Informationen aus dem Aufruf der Ubuntu Startseite in der Standardeinstellung des Werkzeuges. Dabei werden alle implementierten Datenfelder von links nach rechts ausgegeben.
Ein Auszug aus der Hilfe in Dar.6 schlüsselt die Ausgabe auf.

```
FIELDS:

  S: Timestamp(s)  M: Timestamp(ms)  c: Client IP  s: Server IP
  C: Class         Q: Query          T: Type       A: Answer
  t: TTL           n: Count
```

[Dar.6 Erläuterung Datenfelder PassiveDNS]

PassiveDNS kann sowohl IPv4 als auch IPv6 Datenverkehr verarbeiten. Dabei spielt es keine Rolle ob dieser über TCP oder UDP abgewickelt wird. Als Testplattform für dieses Werkzeug wurde Ubuntu 12.04 x64 [40] genutzt.

3.1.2 Tshark

Bei *tshark* handelt es sich um die Konsolenversion des bekannten Werkzeuges *Wireshark* [38]. Mit diesem Analysewerkzeug lässt sich Datenverkehr direkt von der Netzwerkschnittstelle anzeigen, aufzeichnen und abspeichern. Darüber hinaus bietet es zahlreiche Funktionen zur Analyse, wie beispielsweise die Verfolgung von Datenströmen, die Visualisierung von Handshakes, Aufschlüsselung von Datenpaketen sowie eine Export-Funktion, die diverse Formate für aufgezeichnete Netzwerkdaten unterstützt. Darunter auch das gängige *pcap* Dateiformat. Die Dokumentation ist sehr umfangreich gestaltet, da das Werkzeug diverse Funktionen zur Netzwerkanalyse bereitstellt.
Die ursprüngliche Version stammt von Gerald Combs und wurde seitdem durch weltweite Unterstützung stetig weiterentwickelt. Die aktuelle Version vom 04.03.2015 ist 1.12.4. Wireshark ist mittlerweile fester Bestandteil vieler Linux Distributionen wie beispielsweise *Kali Linux* (ehemals Back Track) [43].
Als kompatible Plattformen sind sowohl *Windows* (x64 und x86) als auch Linux Systeme wie *Ubuntu*, *Debian*, *Gentoo*, *Mandriva*, *RedHat* u.v.m. angegeben. Eine vollständige Liste der kompatiblen Betriebssysteme ist in der Anleitung von Wireshark zu finden. Bei Windows Systemen wird zur Aufzeichnung von Netzwerkpaketen zusätzlich noch die Software *WinPcap* benötigt, diese ist aber in den für Windows angebotenen Paketen bereits enthalten. Das Werkzeug lässt sich auch mit Hilfe einer GUI bedienen. Für die Verarbeitung und Analyse von DNS-Datenverkehr werden entsprechende Filter des Werkzeugs verwendet.

```
frame.time||ip.src||ip.dst||class||dns.qry.name||dns.qry.type||dns.resp

Mar 30, 2005 00:47:46.496046000 192.168.170.8   192.168.170.20  0x0001  google.com
Mar 30, 2005 00:49:18.685951000 192.168.170.8   192.168.170.20  0x0001  www.netbsd.org
Mar 30, 2005 00:49:18.734862000 192.168.170.20  192.168.170.8   0x0001  www.netbsd.org
```

[Dar.7 DNS Verkehr tshark nach Filterung]

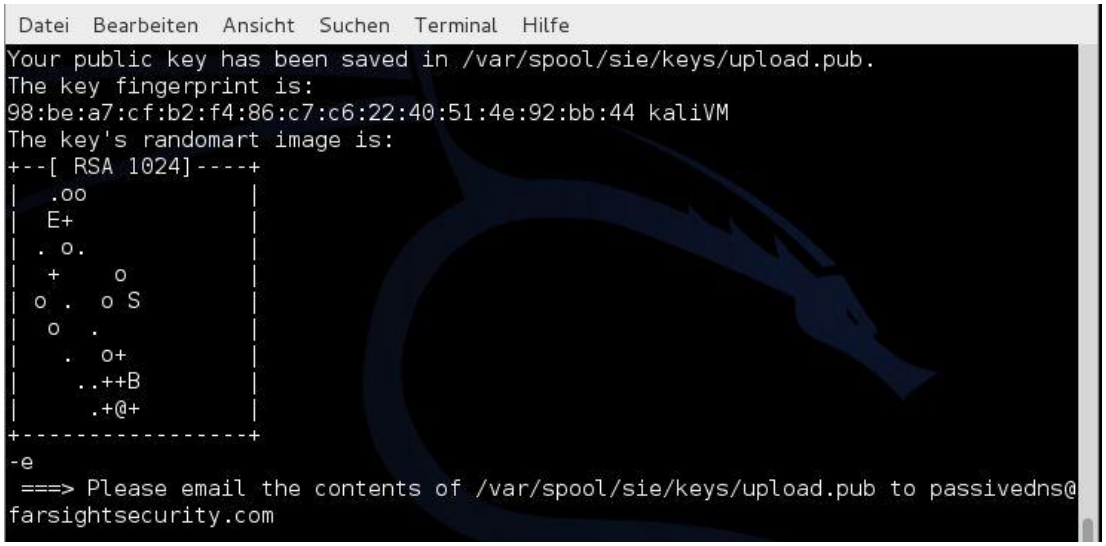
Dar. 7 zeigt DNS Informationen die mittels Filtern extrahiert wurden. Die Hexadezimalen Werte stellen Indizes in Standardtabellen dar. Somit steht der Wert *0x0001* für die Resource Record Klasse *Internet (IN)*.

3.1.3 YAF

YAF [34] ist ein Netzwerkfluss Analyse Werkzeug vom CERT NETSA der Carnegie Mellon University. Es bietet die Funktionalität Flussdaten direkt von der Netzwerkschnittstelle mitzuschneiden, oder aus zuvor aufgezeichneten *pcap* Dateien zu generieren. Die so gewonnenen Daten lassen sich dann von weiteren Werkzeugen verarbeiten, die ebenfalls vom CERT NETSA bereitgestellt werden.
YAF benötigt die Bibliotheken *glib*, *libpcap*, *libfixbuf*, *libairframe* sowie *libyairframe*. Die Bibliotheken *libairframe* und *libyaf* sind bereits im Installationspaket enthalten, die verbleibenden Bibliotheken müssen vor der Installation bei den entsprechenden Herstellern bezogen und auf dem Zielsystem installiert werden.
Die Dokumentation setzt viel Fachwissen voraus und ist aus diesem Grund an manchen Stellen lückenhaft. Teilweise sind die Einträge auch veraltet, was die Einrichtung verzögern kann, da zunächst die aktuellen Parameter recherchiert werden müssen.
Ergänzend zu YAF gibt es diverse Werkzeuge.
Eines davon ist *yafscii* [35]. Es bietet die Funktion, von YAF generierte IPFIX Netzwerkflussdaten im *ASCII* Format ausgeben zu können. Zur weiteren Verarbeitung der aufgezeichneten Netzwerkflussdaten gibt es *super_mediator* [36]. Das Werkzeug enthält zahlreiche Filter, um DNS relevante Informationen extrahieren zu können. Dazu zählen beispielsweise Resource Record Typ, TTL oder der Inhalt des Resource Records. Darüber hinaus bietet es Export-Funktionen für verschiedenen Dateiformate, oder Datenbanken wie *MySQL* oder *PostgreSQL*. Mit Hilfe einer Kombination dieser Funktionen, lässt sich eine Sammlung von DNS-Daten erzeugen. Die einzelnen Komponenten können dabei auch verteilt agieren. So können Informationen aus verschiedenen Netzwerken zentral gesammelt werden.
Ergänzend hierzu gibt es *SiLK [37]*, eine Sammlung von Analysewerkzeugen zur Auswertung von Netzwerkflussdaten. Hier ist zum Beispiel eine *Deep Packet Inspection* Funktion integriert, die Details zu Protokollen oder Nutzlasten von Netzwerkpaketen liefern kann. Diese Informationen können zur zusätzlichen Anreicherung der DNS Daten verwendet werden.

3.1.4 Farsight DNS Sensor

Der PassiveDNS Sensor von *Farsight* [31] ist ein Werkzeug zur ausschließlichen Sammlung von rekursivem DNS Netzwerkverkehr. Dabei werden die Netzwerkflussdaten direkt von der Netzwerkschnittstelle aufgezeichnet und gefiltert. Der Sensor kann über eine verschlüsselte SSH Verbindung mit der Farsight DNSDB verbunden werden. Anschließend kann dieser die Datenbank mit seinen gesammelten (ggf. anonymisierten) Netzwerkdaten ergänzen. Dar. 8 zeigt die Generierung des Fingerprints für die Anmeldung am Farsight Server:

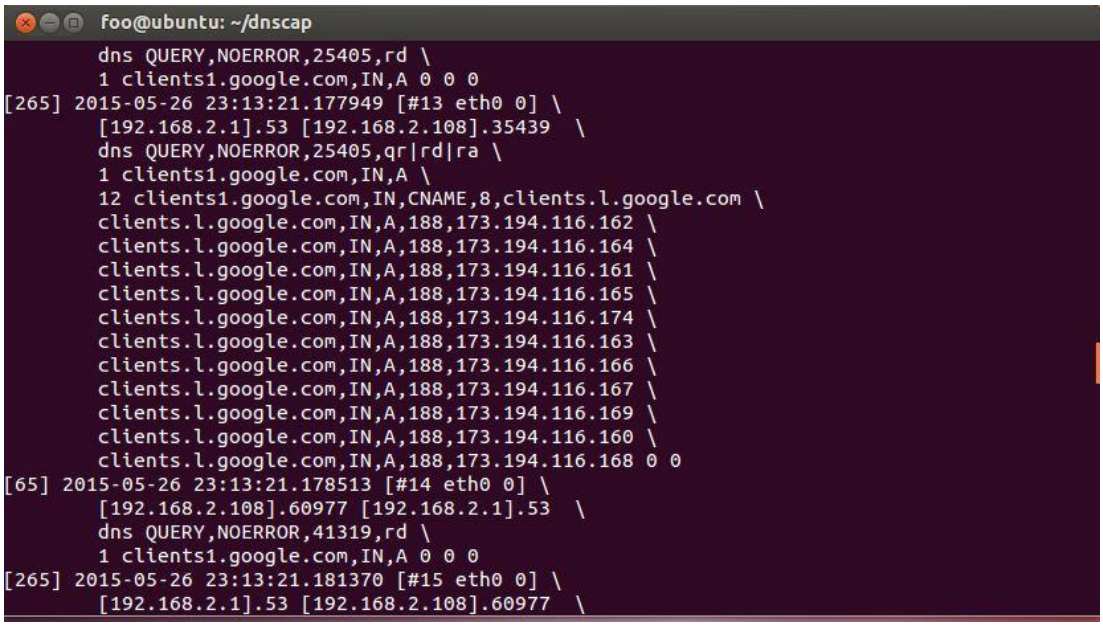


[Dar. 8 Generierung des Fingerprints beim Farsight Passive DNS Sensor]

Der Zufallsfaktor bei der Schlüsselgenerierung wird hier durch Mausbewegungen verbessert. Die Farsight DNS-Datenbank kann zu Analysezwecken genutzt werden, in dem verdächtige Vorfälle mit ihr abgeglichen und bewertet werden können. Der Passive DNS Sensor soll sowohl auf *Debian, RedHat als auch auf FreeBSD* Distributionen (x86 oder x64) lauffähig sein. Die Dokumentation [30] ist entsprechend des Funktionsumfanges sehr kurz gehalten, enthält aber alle notwendigen Informationen um das Werkzeug installieren und in Betrieb nehmen zu können. Es kann sowohl IPv4 als auch IPv6 Netzwerkverkehr verarbeiten. Eine GUI zur Bedienung bringt dieses Werkzeug nicht mit. Es wird empfohlen auf dem Zielsystem einen NTP Client auszuführen. Korrekte Zeitstempel spielen eine wichtige Rolle bei der Datensammlung und der anschließenden Archivierung. Soll ein Ereignis bis zu einem gewissen Punkt zurückverfolgt werden, ist dies nur durch einen lückenlosen Zeitverlauf möglich. Die Konfiguration lässt sich über eine zentrale Datei im Benutzerverzeichnis bearbeiten. Hier können beispielsweise IP-Adressen von Netzwerken angegeben werden, die überwacht werden sollen.

3.1.5 DNSCAP

DNScap [33] von Verisign ist ein Werkzeug zur Aufzeichnung von DNS Netzwerkverkehr im *pcap* Datenformat. Es kann IPv4, als auch IPv6 Datenverkehr über UDP oder TCP verarbeiten und unterstützt dabei auch IP Fragmentierung. Dabei können entweder nur DNS Anfragen, DNS Antworten, oder beides gespeichert werden. Durch das Setzen von Filtern lassen sich bestimmte Ziel oder Quelladressen für sich getrennt betrachten. Eine Zeitsteuerung ist ebenfalls integriert, um das Werkzeug automatisiert starten oder stoppen zu können. Die Dokumentation [32] beinhaltet lediglich eine stichwortartige Erläuterung der Programmparameter sowie kurze Installationshinweise. Es wurden explizit keine kompatiblen Plattformen genannt. Innerhalb der Dokumentation werden aber *Ubuntu* und *FreeBSD* als Beispiele erwähnt.



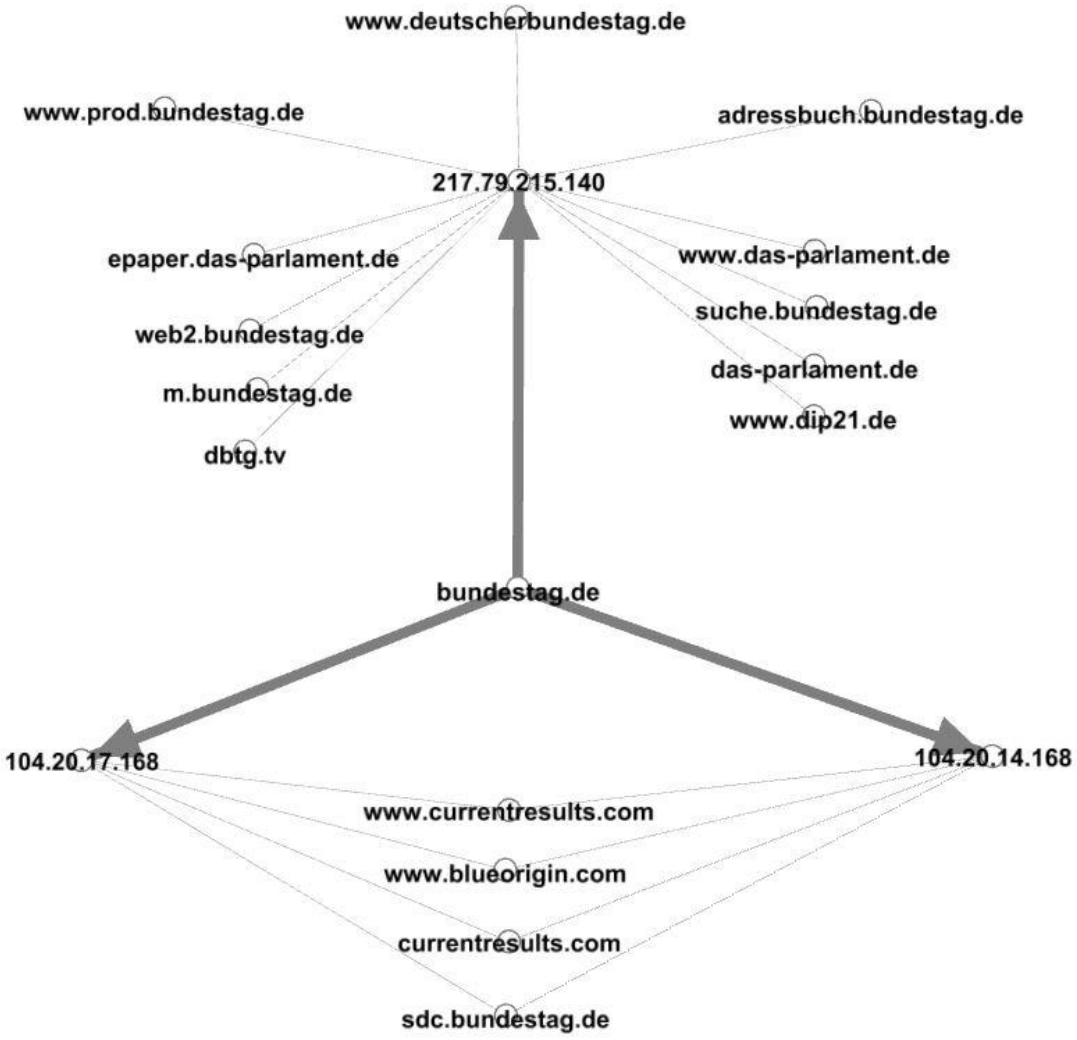
[Dar. 9 Aufzeichnung von DNS-Daten bei Aufruf von Google]

Dar.9 zeigt einen Beispielaufruf von Google. Abgebildet ist hier der anfragte Domainname, gefolgt vom Resource Record Typ und der Klasse. Das Zeilenende bildet die IP-Adresse innerhalb der DNS Antwort. Gut zu sehen ist das Lastenausgleichsystem welches mehrere IP-Adressen zum selben Domainnamen liefert. Bei einem Verbindungsaufbau wählt der Client zufällig einen Eintrag aus.

3.1.6 passiveDNS-client

Der passiveDNS-client [29] von Chris Lee ist ein Analyse- und Visualisierungswerkzeug für DNS-Daten. Es handelt sich hierbei um ein *RubyGem*. Diese lassen sich plattformunabhängig über einen eigenen Paketmanager installieren. Mit dem passiveDNS-client können Abfragen zu Domainnamen, oder IP-Adressen an verschiedene passive DNS-Datenbanken getätigt werden. Hierzu gehören BFK.de, CIRCL, DNSDB (Farsight), Mnemonic, PassiveDNS.cn, PassiveTotal, RiskIQ, TCPIPUtils sowie VirusTotal. Dabei kann auch rekursiv vorgegangen werden, was bei übermäßigem Gebrauch allerdings zu einer Sperrung des entsprechenden Zugangs führen kann. Um diesen Umstand zu vermeiden, empfiehlt der Autor den Parameter für die Tiefe der Funktion nicht über drei zu wählen. In einer gesonderten Konfigurationsdatei können alle Zugänge zu den entsprechenden Datenbanken in Form von API-Keys oder Login Kombinationen hinterlegt werden. Um die mitgelieferte Anbieterliste durch eigene Anbieter zu erweitern, wird in der Dokumentation ein Code Skelett vorgegeben, mit dem sich eigene Datenbankadapter programmieren lassen.

Eine weitere Funktion des Werkzeuges ist die Ausgabe des Anfrageergebnisses in ein visuelles Graphen Format. Hierbei kann zwischen dem *GDF-*, dem *graphviz-*, oder dem *Graphml*-Format gewählt werden. Weitere Ausgabeformate, wie *CSV*, *XML*, *YAML*, *JSON* oder *ASCII* werden ebenfalls unterstützt. Die Ausgabe soll zum RFC Entwurf *Passive DNS - Common Output Format* kompatibel sein, dies konnte allerdings noch nicht bestätigt werden, da es derzeit noch keine Implementierungen dieses Entwurfes gibt. Zur Informationsbereitstellung können neben der Konsole auch *sqlite* Dateien verwendet werden. Dies ist bei größeren Datenansammlungen von Vorteil.



[Dar. 10 Visualisierte Ausgabe der Anfrage bundestag.de an VirusTotal)

Dar. 10 zeigt die visualisierte Ausgabe für die Anfrage *bundestag.de.* Dabei wurde der Parameter für die Graphenausgabe gesetzt. Die Ausgabe wurde in eine Datei umgeleitet. So lässt sich diese in einem beliebigen Graphen Editor öffnen und bearbeiten. Hierzu wurde *Gephi* 0.8.2 beta [42] verwendet. Gut zu sehen ist das Lastenverteilungssystem mit 2 Server IP-Adressen, sowie diverse Domainnamen. Betrachtet man Domainnamen wie *dip21* für sich, würde man diese nur schwer in Verbindung mit dem Bundestag bringen. Dieser Graph zeigt allerdings anschaulich, dass sie Teil der Zonendatei ist. Diese Netzwerktopografischen Informationen können eine Analyse begünstigen.

3.2 Entwicklung einer Bewertungsmatrix für passive DNS Werkzeuge

Zunächst werden Kriterien festgelegt, die im Zuge einer Analyse genauer beleuchtet werden können. Die Untersuchungsergebnisse werden dann zur besseren Visualisierung zusätzlich in einer Matrix dargestellt. Diese unterliegt einem mehrstufigen Bewertungssystem. Dabei werden in jeder Kategorie Punkte von eins bis vier vergeben. Vier steht für die beste Bewertung, eins entsprechend für die schlechteste. Um bestimmte Aspekte stärker hervorzuheben, unterliegt die Matrix einer Gewichtung. Dabei werden die Zahlen eins bis drei als Multiplikator für die Punkte verwendet. Die Maximalpunktzahl liegt so bei 64 Punkten. Bewertet werden die folgenden Kriterien:

Dokumentation:
Hierbei geht es um die Qualität der verfügbaren Dokumentation. In die Bewertung fließen jegliche Formen von Dokumentationen ein, wie Readme, Textdateien, Wiki, Foren, oder die integrierte Hilfe der Werkzeuge. Die Verständlichkeit wird ebenfalls bewertet, da Softwareautoren oft Dinge voraussetzen, die Außenstehenden nicht geläufig sein können.

Support:
Bei diesem Kriterium spielt der Kontakt zu Experten eine Rolle. Es wird bewertet, ob es Möglichkeiten gibt, die Entwickler, oder mit dem Werkzeug vertraute Personen, direkt erreichen zu können, um Unklarheiten zu beseitigen oder Fragen zu stellen. Hierfür bietet das Internet eine Vielzahl von Möglichkeiten wie E-Mail, Foren, oder Mailinglisten.

Performance:
Datendurchsatz und Speicherverbrauch sind Thema dieses Kriteriums. Um gleiche Bedingungen für alle Werkzeuge zu schaffen, werden *pcap* Dateien verwendet die aufgezeichnete Netzwerkdaten enthalten. Diese sind für jedes Werkzeug identisch. Bei der Verarbeitung werden die Laufzeit und der Speicherverbrauch gemessen.

Der passiveDNS-client ist als Auswertungswerkzeug in einer Sonderposition. Hier wird keine Funktionalität zum Mitschneiden von Netzwerkverkehr benötigt. Zu analysierende Daten können über *sqlite* Dateien, oder die Konsole eingelesen werden.

Alle anderen Werkzeuge implementieren das Mitschneiden von Daten über die Netzwerkschnittstelle. Dateien im pcap Format können eingelesen und verarbeitet werden. Das Exportieren von DNS-Daten wird ebenfalls von allen Werkzeugen unterstützt.

3.3 Werkzeuganalyse

3.3.1 Dokumentation

Zur Bewertung der jeweiligen Werkzeug Dokumentation, wurden die Kriterien Einrichtung, Verwendung, Hilfe sowie Verständlichkeit gewählt. Jeder dieser drei Aspekte erhält seinen eigenen Gewichtungsfaktor. Die Kategorien Einrichtung und Verwendung werden hervorgehoben. Soll ein Werkzeug zum ersten Mal installiert werden, so spielt der entsprechende Teil in der Dokumentation eine essentielle Rolle. Zunächst sollte hier auf eventuelle Abhängigkeiten und Vorbedingungen hingewiesen werden, damit diese vor der Installation überprüft und hergestellt werden können. Hierzu gehören beispielsweise Bibliotheken, die in bestimmten Versionen vorhanden sein müssen, oder Einträge in systemeigene Konfigurationsdateien. Alle notwendigen Schritte sollten beschrieben werden, im besten Fall mit einer Musterinstallation. Dabei dürfen keine Zwischenschritte fehlen, die bei nicht Beachtung unerwünschte Nebeneffekte auslösen können.

Das Kriterium Verwendung ist nach einer erfolgreichen Installation mit gleichem Stellenwert zu betrachten. Es muss klar sein, welche Eingaben das Werkzeug benötigt und welche Ausgaben es zurückgibt. Beispielhafte Szenarien, oder Testaufrufe mit vorherbestimmtem Verhalten können die gewünschten Informationen liefern.
Ergänzend hierzu soll auch die Werkzeug Hilfe sein. Sie wird in der Regel mit dem Parameter *–h, -?* oder *-help* aufgerufen und soll eine Übersicht der Werkzeugparameter liefern. Hier spielt die korrekte Beschreibung der Parameter und ihrer Einheiten eine wichtige Rolle, um Fehlbedienungen vorzubeugen. Die Übersichtlichkeit soll bei einer großen Anzahl verschiedener Parameter nicht vernachlässigt werden. Klare Gliederungen helfen, sich in der Hilfe zurechtzufinden.
Der letzte Punkt ist die Verständlichkeit der Beschreibungen. Autoren von Werkzeugen können dazu neigen, notwendige Informationen als selbstverständlich anzusehen. Im Zuge ihrer Arbeit haben sie sich intensiv mit einem Thema beschäftigt und es kann Ihnen schwer fallen, die nötige Distanz zum Themengebiet wiederzufinden. Die Dokumentation sollte so formuliert sein, dass auch außenstehende Benutzer erfassen können was gemeint ist. Fachtermini sind bei Bedarf auszuführen, um die Verständlichkeit zu erhöhen.

Innerhalb der Dokumentation von PassiveDNS fehlten die beiden ausschlaggebenden Punkte Einrichtung und Verwendung. Man muss hier selbstständig erkennen, dass *autotools* verwendet wurden und die Installation nach dem entsprechenden Schema durchführen. Das Thema Verwendung wurde in der Dokumentation ebenfalls nicht aufgegriffen. Beispielaufrufe oder Erläuterungen zur Verwendung waren nicht vorhanden. Drei typische Beispiele aus der Praxis sollen zeigen, zu welchen Zwecken sich das Werkzeug nutzen lässt. Die Hilfe beschreibt ausführlich alle Informationen zu Ausgabe- und Eingabeparametern. Hier werden auch Fehlercodes beschrieben. Es wurden keine

Fachbegriffe genutzt, die eine weiterführende Erklärung notwendig machen. Somit kann die Verständlichkeit der vorhandenen Dokumentation als gut bewertet werden.

Wireshark besitzt, wie zuvor beim Punkt Support festgestellt, eine sehr ausführliche Dokumentation. Die Einrichtung wird auf verschiedenen Betriebssystemen beschrieben. Zur Verwendung werden zahlreiche Beispiele genannt. Die Hilfe des Werkzeuges selbst enthält viele Details zu Filtern, Datenverarbeitung sowie Ein- und Ausgabeparametern. Der Dokumentationsstil ist einfach und verständlich gehalten. Auf Fachbegriffe wird im Wiki ausführlich eingegangen.

Die Dokumentation von YAF erwähnt bei der Einrichtung notwendige Abhängigkeiten. Es werden die Bibliotheken *glib* (ab 2.6.4) *libpcap* sowie *libfixbuf* (ab 1.0.0) benötigt.
Weiterhin wird beschrieben, wie das Werkzeug mit Optionen, wie beispielsweise der tieferen Paketanalyse, kompiliert werden kann. Es gibt verschiedene Tutorials zur Verarbeitung von *pcap* Dateien, Export Funktionen, oder der Verwendung mit zusätzlichen Werkzeugen wie Silk oder *super_mediator*. Die Hilfe ist in die Kategorien *log*, *pcap*, *application* und *export* unterteilt. Die Beschreibungen der Parameter sind für dessen Gebrauch ausreichend detailliert ausgeführt. Die Dokumentation setzt ein gewisses Fachwissen über die Netzwerkflussanalyse voraus. Aus diesem Grund wird die Verständlichkeit nur mit einem gut bewertet.

Für den Passive DNS Sensor von Farsight werden Installationspakete für *Debian*, *Ubuntu*, sowie *RedHat* innerhalb der Dokumentation angegeben. Die Installation kann bei diesen Distributionen also über den integrierten Paketmanager erfolgen. Weiterhin werden die Parameter der Konfigurationsdatei thematisiert. Das Werkzeug wird als Systemdienst installiert und entsprechend bedient. Die dazugehörigen Parameter *start*, *stop* sowie *restart* werden erläutert. Die Option *force-reload* wird innerhalb der Hilfe ergänzt. Mit dieser Option wird lediglich die Konfiguration des Dienstes neu geladen, der Dienst wird dazu nicht beendet. Durch *restart* wird der Dienst gestoppt und anschließend wieder gestartet. In dieser Zeit werden keine Daten aufgezeichnet. Diese Information ist nicht in der Dokumentation aufgeführt. Die Verwendung wird durch viele Beispielaufrufe verdeutlicht. Die Verständlichkeit leidet nicht durch übermäßigen Gebrauch von Fachtermini.

Der Punkt Einrichtung wurde bei DNScap mit unzureichend bewertet, da auf ihn in der Dokumentation nicht eingegangen wird. Das Werkzeug wird, wie PassiveDNS, mittels *autotools* kompiliert und installiert. Dies ist allerdings erst nach einer genaueren Betrachtung des *Repositories* ersichtlich. Ähnlich verhält es sich bei der Verwendung. Es gibt weder Beispielaufrufe noch Szenarien die thematisiert werden. Die Hilfe ist in einer kurzen und einer ausführlichen Form verfügbar. Für die ausgedehnte Hilfe gibt es den Parameter *-?*. Da die kurze Hilfe für sich keinerlei Erläuterungen enthält, empfiehlt es sich auf die längere Form zurückzugreifen. Diese beschreibt alle relevanten Details der

Werkzeugparameter. Die Verständlichkeit wird mit gut bewertet, allerdings umfasst die Dokumentation, die Hilfe ausgenommen, nur drei Punkte zum kompilieren mit *libbind*.

Der passiveDNS-client wird über den Paketmanager *RubyGems* installiert. Die notwendigen Schritte hierfür werden mit Hilfe von Beispielaufrufen in der Dokumentation geschildert.
Die Einrichtung der Zugangsdaten zu den verschiedenen passive DNS-Datenbanken wird mit einer Musterkonfiguration erläutert. Es werden zudem URLs genannt, um die notwendigen Zugangsdaten anfordern zu können. Die Hilfe zeigt die Parameter der einzelnen Datenbanken, die frei miteinander kombiniert werden können. Zudem werden Ausgabeparameter und Informationen zu rekursiven Anfragen thematisiert.
Ein Codeskelett, um weitere Datenbankanbieter hinzufügen zu können, wird am Ende der Dokumentation vorgegeben. Durch eine Vielzahl von Beispielen und eine klare Gliederung, wirkt die Dokumentation insgesamt sehr verständlich.

3.3.2 Support

Darstellung 11 zeigt eine Gegenüberstellung der Kommunikationsmöglichkeiten mit den Entwicklern der jeweiligen Werkzeuge. Ein „X“ innerhalb einer Tabellenspalte gibt an, ob die entsprechende Kommunikationsform angeboten wird. E-Mail steht hierbei für die Möglichkeit eines direkten Mailverkehrs, im Gegensatz zu Mailinglisten die öffentlich als Gruppe geführt werden. Um die Existenz der E-Mail Adressen zu überprüfen, wurde jeweils eine Testmail verschickt. Alle Testmails sind ohne Fehlermeldung im entsprechenden Postfach eingegangen. GitHub steht hier stellvertretend für alle Funktionen die das Portal zur Kommunikation bietet, wie beispielsweise Kommentarfunktionen oder Tickets. Forum meint an dieser Stelle eine Diskussionsplattform, die nicht zwingend vom Hersteller selbst betrieben wird, sondern auch Plattformen externer Betreiber mit einschließt. *Q & A* ist ein System, bei dem Benutzer öffentlich Fragen stellen können. Die Fragen werden darauf von Entwicklern, ebenfalls öffentlich, beantwortet. Später kann aus der so entstehenden Sammlung ein statisches FAQ erstellt werden.

	E-Mail	**GitHub**	**Forum**	**Q & A**	**Mailingliste**
Gewichtung	20 %	20 %	20 %	20%	20%
PassiveDNS	X	X			
Wireshark	X	X	X	X	X
YAF	X				X
Farsight Passive DNS Sensor	X	X			
DNSCAP	X	X			X
PassiveDNS-client	X	X			

[Dar. 11 Supporttypen]

Die wenigsten Möglichkeiten bietet an dieser Stelle das Werkzeug YAF, da es lediglich eine E-Mail Adresse angibt und kein öffentliches Repository verwendet. Im Gegensatz hierzu stehen PassiveDNS, der Passive DNS Sensor von Farsight, DNScap sowie der passiveDNS-client. Diese Werkzeuge werden alle samt auf der Plattform GitHub bereitgestellt. Sie bieten so auch die Möglichkeit aktiv an den Projekten mitzuwirken.
Wireshark bietet zusätzlich noch die Möglichkeit eines Diskussionsforums, pflegt einen interaktiven Q & A Bereich und bietet eine Mailingliste an. Man muss an dieser Stelle allerdings sagen, dass Wireshark ein sehr vielseitiges, weit verbreitetes Werkzeug ist, von dem nur ein kleiner Teil des Funktionsumfanges [39] für die DNS Analyse benötigt wurde. Es ist seit 2006 verfügbar und verfügt über eine große Community. Dieser Umstand hat einen positiven Einfluss auf die Qualität des Supports.

3.3.3 Performance

Um die Performance der verschiedenen Werkzeuge erfassen zu können, werden zunächst messbare Kriterien aufgestellt. Ein Merkmal für die Performance ist die Zeit, die gebraucht wird, eine bestimmte Anzahl von DNS-Datenpaketen zu verarbeiten. Ein weiteres der Speicherverbrauch innerhalb dieser Zeit. Um diese beiden Kriterien zu messen, wurde innerhalb einer virtuellen Maschine eine Linux Testumgebung eingerichtet.

Die Tests wurden innerhalb des *VMWare* Players 7.0.0 build-2305329 [45] durchgeführt. Der virtuellen Maschine wurde ein CPU Kern sowie 1GB Ram zugewiesen. Als Gastsysteme wurden *Ubuntu 14.04 LTS x64* [40] sowie *Kali Linux 1.1.0a x64* [43] mit aktuellen Paketquellen vom 01.07 eingesetzt. Das Hostsystem ist mit einem AMD Phenom II X4 mit 3,20 GHz und 16GB Ram Speicher ausgestattet. Als Mainboard wird ein Asus M4A79XTD Evo verwendet. Das Hostsystem wird mit Windows 7 x64 SP1 betrieben. Die Messung der Performance wird durch die Benchmark Funktion des actor-frameworks [41] durchgeführt. Zur Automatisierung der Vorgänge wurde ein *Shellscript* angefertigt. In Dar. 12 werden relevante Auszüge aus diesem Script aufgeschlüsselt:

```
24 while (($x < $n ))
25 do
26  #Test
27  sudo ./caf_run_bench 1000 5 0 laufzeit
speicher/speicher_$x ~/passiveDNS/src/passiveDNS -r
DNS_1000.pcap
28  #Caches leeren
29  sudo sync && echo 3 | sudo tee /proc/sys/vm/drop_caches
```

[Dar. 12 Shellscript Test_Speicher_Laufzeit Testschleife]

Zeile 25 zeigt den Aufruf der Testfunktion des actor-framworks. Von links nach rechts haben die Parameter folgende Bedeutung:

„1000“ steht für die ID Benutzers der die zu testende Anwendung ausführt. „5“ beschreibt die maximale Laufzeit des zu testenden Prozesses in Sekunden. „0“ steht für das Polling Intervall der Speicherauslastung. Die Ausgaben der Messung werden jeweils in die Textdateien „laufzeit“ und „speicher“ geschrieben. Jeder Testdurchlauf erzeugt eine eigene Speicher-Datei, die entsprechend des Durchlaufes benannt wird.

```
69   6   184
70   6   184
71   7   184
72   7   204
```

[Dar. 13 Auszug aus einzelner Speicher Datei 16 nach Test von PassiveDNS]

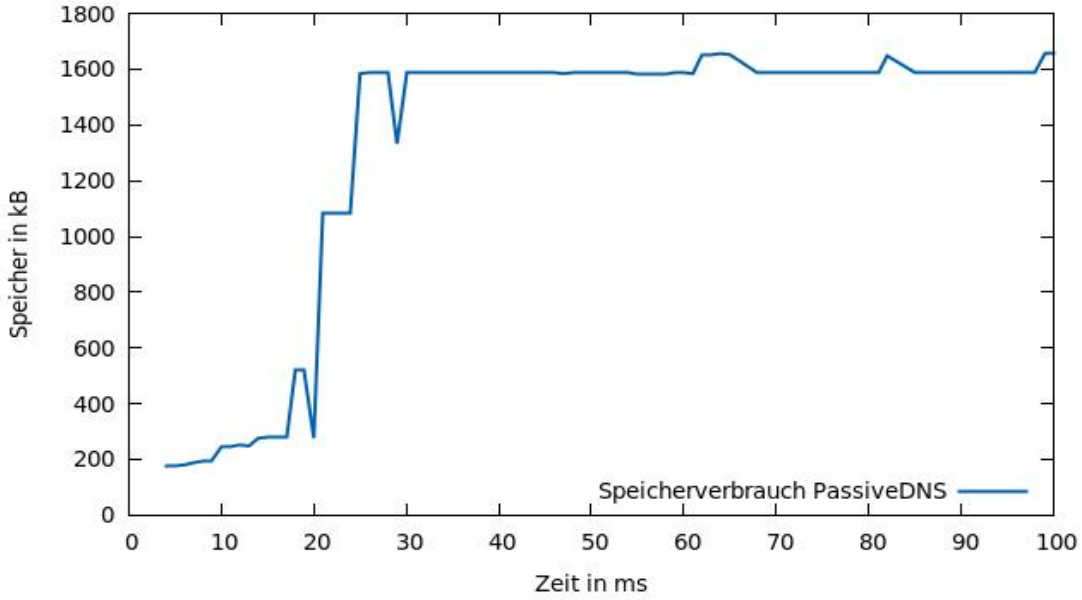
Dar. 13 zeigt, dass die Datei in 2 Spalten aufgeteilt ist. Jede Zeile beginnt mit der Zeit in ms, die seit Aufruf des Werkzeuges vergangen ist, gefolgt vom Speicherverbrauch in kB. Da die Messungen teils schwanken, werden sie im weiteren Verlauf zur Auswertung aufbereitet. Um die zahlreichen Dateien einfacher zusammenfassen zu können, werden sie in ein Unterverzeichnis namens „speicher“ geschrieben. Das Ende des Befehls bildet das zu testende Werkzeug mit den entsprechenden Parametern zum Einlesen einer pcap Datei.
Die zu verarbeitende pcap Datei wurde zuvor mit Hilfe von Wireshark erstellt. Sie enthält genau 1000 DNS Einträge. Dieser Wert bietet sich an, da ein Durchschnittswert aus 100 Messungen berechnet werden soll, um Messungenauigkeiten auszugleichen. Für alle Performancemessungen wurde dieselbe Datei verwendet.
Nach jedem Testdurchlauf werden in Zeile 27 die systemeigenen Caches geleert, um eine Verfälschung der Laufzeit durch Caching zu verhindern. Durch eine Schleife kann die Anzahl der Durchläufe per Variable angepasst werden.
Nach der letzten Schleifeniteration wird die Datenauswertung angestoßen.
Dar. 14 zeigt den entsprechenden Ausschnitt des Scripts.

```
34      #Dateien zusammenfassen
35      cat speicher/* > speicher/speicher_zusammen
36      #Daten sortieren
37      sort -n speicher/speicher_ges > speicher/speicher_ges_sort
38      sort -n laufzeit > laufzeit_sortiert
39      #Median für jeden ms Wert berechnen
40       cat  speicher/speicher_ges_sort  |  python3  mediane.py  >
speicher/speicher_median
```

[Dar. 14 Shellscript Test_Speicher_Laufzeit Auswertung]

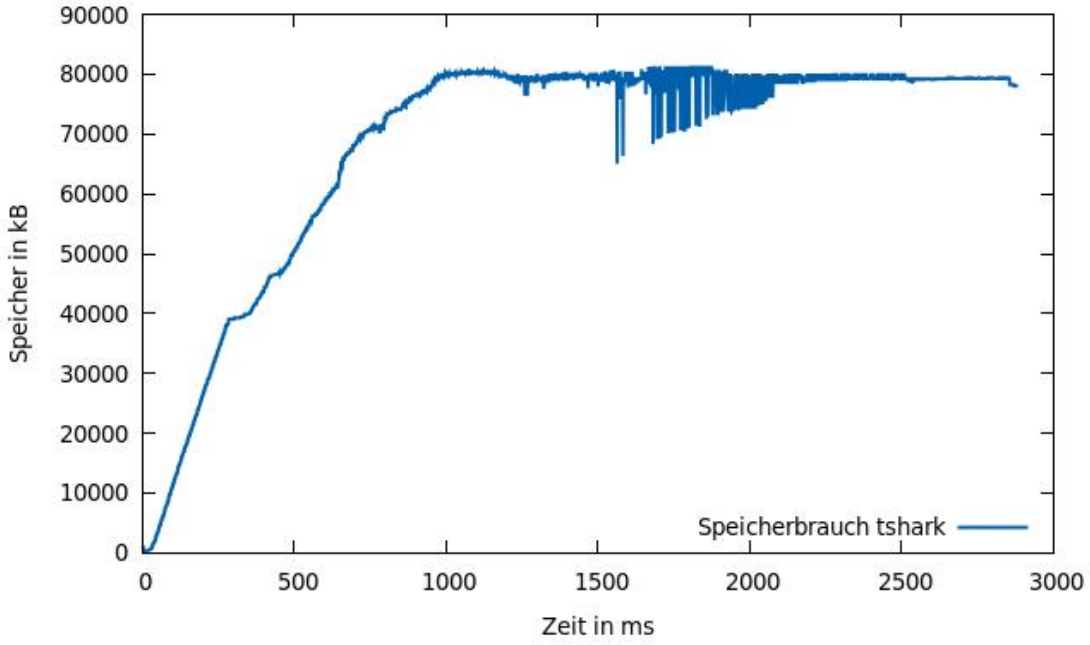
Zeile 35 fasst den Inhalt der einzeln angelegten Dateien in einer Gesamtdatei zusammen. Diese wird im Anschluss numerisch sortiert und an ein Python Script übergeben.
Das Script bestimmt aus den Messdaten für jeden Zeitwert den Median und schreibt die Ergebnisse in eine neue Datei.
Die Laufzeit jedes einzelnen Durchganges in ms wird in die Datei *laufzeit* geschrieben.
Maximum, Minimum sowie Durchschnitt werden sowohl für den Speicherverbrauch als

auch für die Laufzeit bestimmt. Die Ergebnisse werden, nach dem letzten Schleifendurchlauf, in eine Auswertungsdatei übertragen. Die so gewonnenen Werte werden mit Hilfe von *Gnuplot* [44] visualisiert. Das gesamte Shellscript ist dem Anhang dieser Arbeit beigefügt. Im Folgenden werden die Auswertungen der Testdurchläufe für die einzelnen Werkzeuge dargestellt.



[Dar. 15 Speicherverbrauch PassiveDNS]

Zum Test wurde PassiveDNS mit dem *–r* Parameter aufgerufen, um eine pcap Datei einzulesen. Wie in Dar. 15 gezeigt, bewegt sich der Speicherverbrauch bei diesem Test zwischen 176kB und 1.656kB. Im Durchschnitt werden 1.308kB Speicher benötigt. Für die Verarbeitung von 1.000 Netzwerkpaketen wurden im Schnitt 46 ms gebraucht. Das entspricht einem Paketdurchsatz von etwa 22 Paketen/ms.



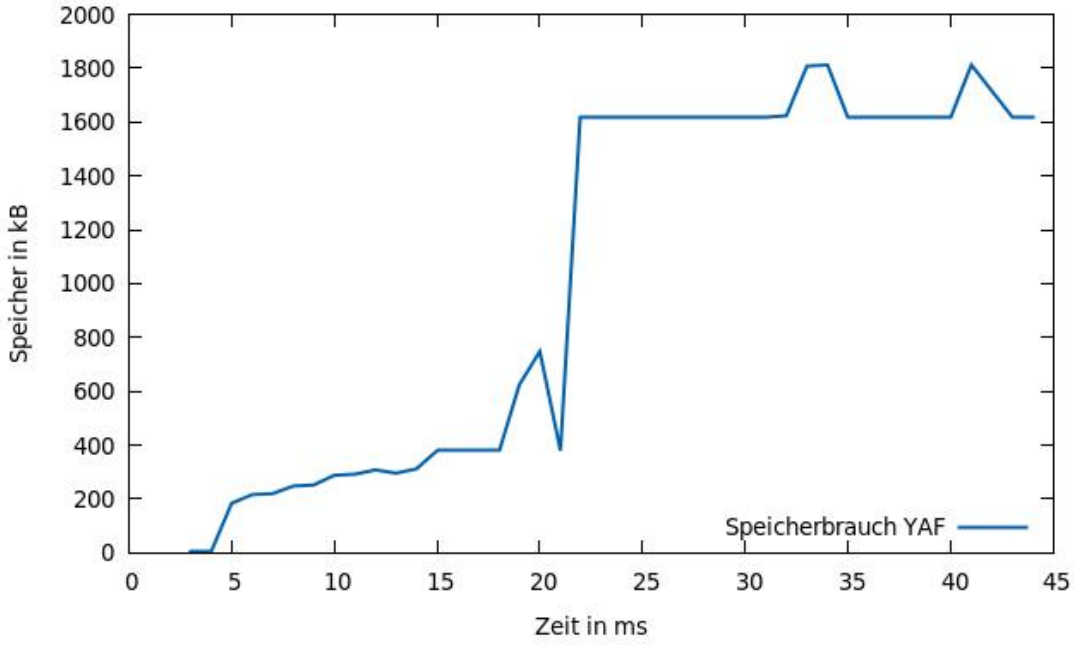
[Dar. 16 Speicherverbrauch tshark]

tshark wurde wie folgt aufgerufen (Dar.17):

```
27  sudo ./caf_run_bench 1000 5 0 laufzeit speicher/speicher_$x
tshark -r DNS_1000.pcap -T fields -e frame.time -e ip.src -e ip.dst
-e DNS.qry.class -e DNS.qry.name -e DNS.qry.type -e DNS.resp.addr -e
DNS.resp.ttl > DNS.txt
```

[Dar. 17 Aufruf von tshark]

Die –e Parameter stehen jeweils für einen Filter [39]. Mit Hilfe dieser Filter werden die relevanten Informationen aus der pcap Datei extrahiert. Um die Vergleichbarkeit zu gewährleisten, wurden dieselben Informationen gewählt die auch PassiveDNS in seiner Standard Einstellung extrahiert. Dabei handelt es sich um:
Zeitstempel, DNS-Cient, DNS-Server, RR-Klasse, Anfrage, Anfragen Typ, Antwort des DNS-Servers, TTL.
Dar. 16 zeigt das Minimum des Speicherverbrauchs bei 4kB, das Maximum erreichte 19.448kB. Durchschnittlich wurden 66.810kB Speicher verbraucht. Die Testdatei wurde im Schnitt innerhalb von 1.304ms verarbeitet. Somit wird ein Datendurchsatz von rund 1 Paket/ms erreicht. Der Speicherverbrauch steigt mit der Laufzeit an.



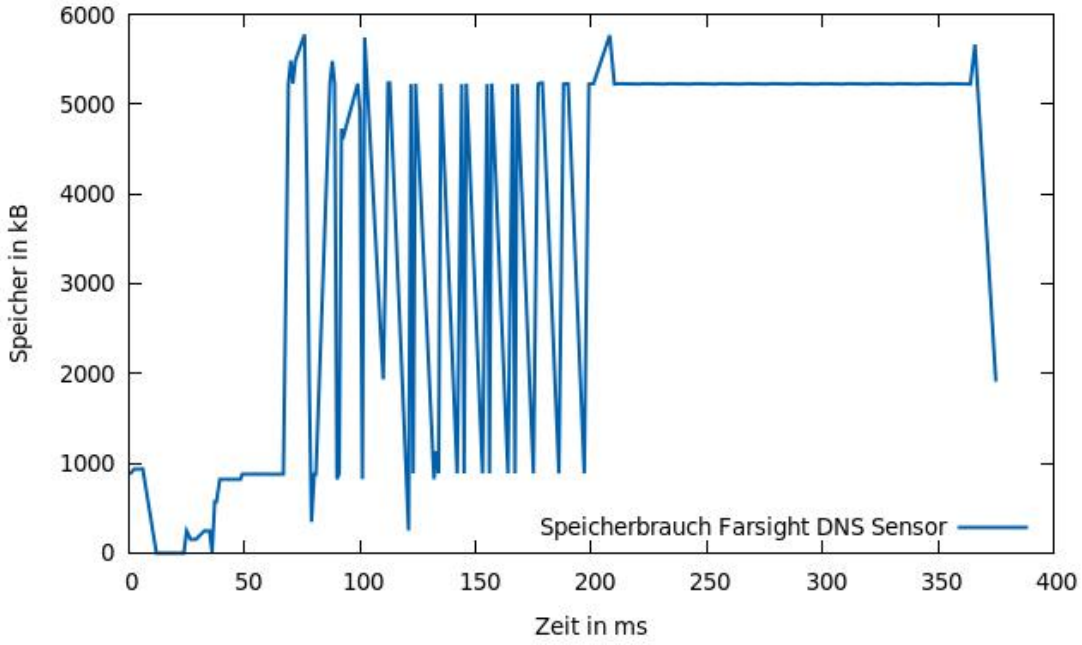
[Dar. 18 Speicherverbrauch YAF]

Der Testaufruf von YAF enthält die Parameter für das Einlesen einer Datei, sowie die Ausgabe im *IPFIX* Datenformat (Dar. 19).

```
27  sudo ./caf_run_bench 1000 5 0 laufzeit speicher/speicher_$x
/usr/local/bin/yaf --in DNS_1000.pcap --out DNS.yaf
```

[Dar. 19 Aufruf von YAF]

Die schnellste Instanz von YAF benötigte 22ms die langsamste 56ms. Im Durchschnitt werden 31ms für die 1.000 Testdatenpakete gebraucht. Das entspricht rund 32 Paketen/ms. Der Speicherverbrauch bewegt sich in Dar. 18 zwischen 4kB Minimum nach Programmstart und 1620kB Maximum gegen Programmende. Im Schnitt verbraucht YAF 1027kB Speicher. Der Speicherverbrauch steigt mit der Laufzeit an.



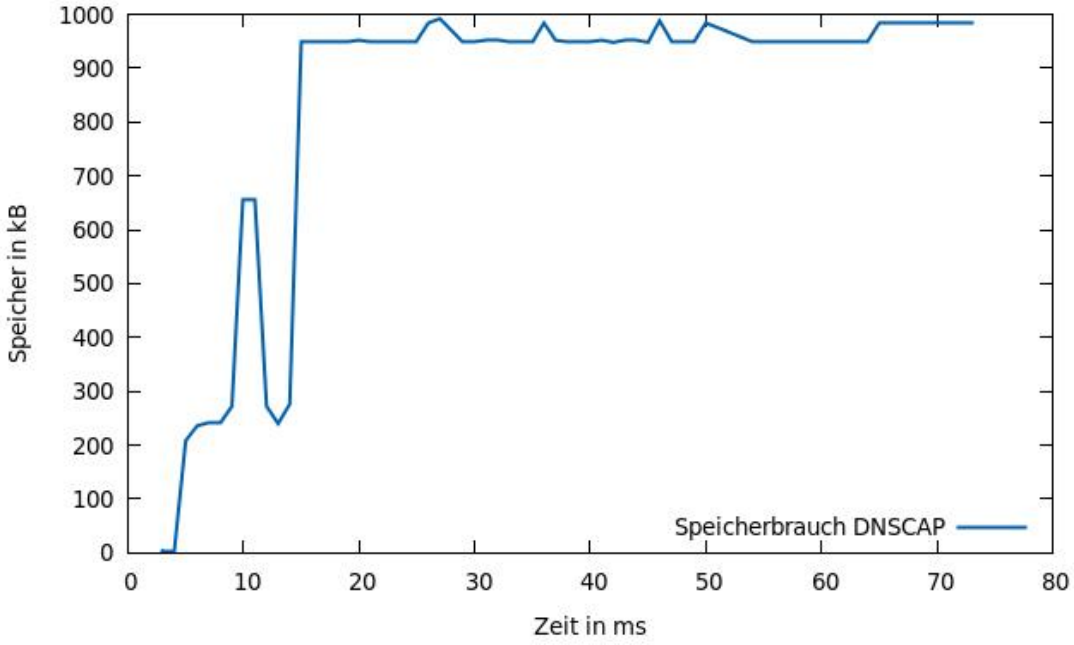
[Dar. 20 Speicherverbrauch Farsight DNS Sensor]

Der Farsight DNS Sensor wurde wie folgt aufgerufen (Dar. 21).

```
27  sudo ./caf_run_bench 1000 5 10 laufzeit speicher/speicher_$x
./nmsgtool -p dns_1000.pcap -V ISC -T dnsqr -w DNS.farsight
```

[Dar. 21 Aufruf Farsight DNS Sensor]

Der kürzeste Durchlauf dauerte 71ms,der langsamste 377ms. Im Durchschnitt wurden 95ms für die 1.000 Testdatenpakete gebraucht. Das entspricht rund 10 Paketen/ms. Der Speicherverbrauch bewegt sich zwischen 4kB Minimum nach Programmstart und 5664kB Maximum gegen Programmende. Die Speicherauslastung schwankt sehr stark im Bereich 60ms bis 200ms. Zum Ende der Laufzeit sinkt der Speicherverbrauch stark ab. Im Schnitt werden 2943kB Speicher verwendet.



[Dar. 22 Speicherverbrauch DNSCAP]

Die Parameter des Aufrufes für DNSCAP sind *–r* für das Einlesen der pcap Datei sowie *–w* für die Ausgabe nach der Verarbeitung (Dar. 23).

```
27  sudo ./caf_run_bench 1000 5 0 laufzeit speicher/speicher_$x
~/DNScap/DNScap -r DNS_1000.pcap -w
```

[Dar. 23 Aufruf von DNSCAP]

Dabei zeigt Dar.22 einen Speicherverbrauch zwischen 172kB und 984kB. Im Durchschnitt werden 816kB erreicht. Die Laufzeit betrug im Schnitt 22ms, sie bewegte sich von 15ms bis hin zu 71ms und stieg mit der Laufzeit an. Daraus ergibt sich ein Paketdurchsatz von rund 45 Paketen/ms.

	Paketdurchsatz	Speicher Ø
Gewichtung	50 %	50 %
PassiveDNS	22 Pakete /ms	1.308 kB
Wireshark	1 Paket/ms	66.810 kB
YAF	32 Pakete/ms	1027 kB
Farsight DNS Sensor	10 Pakete/ms	2943 kB
DNSCAP	45 Pakete/ms	816 kB

[Dar. 24 Zusammenfassung Messergebnisse]

Beide Performance Kriterien wurden gleich gewichtet. DNSCAP liefert in beiden Kategorien das beste Resultat. Es kann die meisten Datenpakete in einer Zeiteinheit verarbeiten. Dabei weist es den geringsten Speicherverbrauch auf. YAF und PassiveDNS liegen im Mittelfeld. Wireshark liefert im direkten Vergleich einen deutlich geringeren Durchsatz sowie einen signifikant größeren Speicherverbrauch.

3.4 Stärken

Alle untersuchten Werkzeuge werden mit Hilfe von *autotools* kompiliert. Dies ermöglicht eine Installation auf unterschiedlichen Linux Systemen. Abhängigkeiten werden hierbei automatisiert überprüft und können individuell für jedes System gelöst werden. Diese Lösung ist, gegenüber fertig kompilierten Paketen für einzelne Plattformen, sehr flexibel.
Ein weiteres positives Merkmal ist die Gestaltung der integrierten Hilfe. Die Beschreibungen sind logisch gegliedert, und bei größerem Informationsaufkommen sinnvoll unterteilt. Die Einheiten und Datentypen wurden klar definiert und lassen wenig Interpretationsspielraum. Damit werden ungewollte Ausgaben, oder Fehlbedienungen auf ein Minimum reduziert. Ein weiteres gemeinsames positives Merkmal ist die Softwarelizensierung. Der Programmcode jedes Werkzeuges wurde unter der Open Source GNU Public License [16] veröffentlicht. Der Programmcode ist öffentlich einsehbar. Dies bietet den großen Vorteil einer breiten Öffentlichkeit, die sich an der Entwicklung beteiligen kann. Auch können individuelle Anpassungen vorgenommen werden, sollte es die Situation erfordern. Der Programmcode kann in eigene Projekte eingebunden und seine Funktionalität vergrößert werden. Mögliche Fehler werden eher gefunden und können schneller behoben werden. Gleichzeitig werden die Werkzeuge kostenlos angeboten.

3.5 Schwächen

Die Dokumentationen der untersuchten Werkzeuge sind teilweise lückenhaft oder setzen tiefer greifendes Fachwissen voraus. Veraltete Befehle oder Programmparameter müssen zunächst als solche erkannt werden um den entsprechenden Schritt erfolgreich abschließen zu können. Dies erfordert zusätzliche Recherche und damit auch Zeit. Einen weiteren Aspekt stellt das Fehlen von Diskussionsplattformen dar. Dies ist zwar nicht unbedingt den Autoren der Werkzeuge geschuldet, allerdings wäre der Austausch von Erfahrungsberichten auf diesem Gebiet sehr hilfreich. Derartige Plattformen könnten, unter stetiger Moderation, gleichzeitig als gute Ergänzung zur Dokumentation dienen.

3.6 Anwendung der Bewertungsmatrix

Dar. 25 zeigt die allgemeine Bewertungsmatrix in die alle Ergebnisse der Werkzeuganalysen eingeflossen sind.

	Zusatzwerkzeuge	Performance	Support	Doku: Verständlichkeit	Doku: Verwendung	Doku Einrichtung	Doku: Hilfe	
Gewichtung	2	3	1	2	3	3	2	**Gesamt**
PassiveDNS	2	2	2	4	1	1	4	**34**
tshark	2	1	4	4	4	4	4	**51**
YAF	4	3	2	3	4	4	4	**57**
Farsight Passive DNS Sensor	1	1	2	4	2	4	4	**41**
DNScap	1	4	3	2	1	1	4	**35**
passivedns-client	1		2	4	4	4	4	**44**

Bewertung		Gewichtung	
sehr gut	4	sehr wichtig	3
gut	3	mittel	2
mangelhaft	2	neutral	1
unzureichend	1		

[Dar. 25 Bewertungsmatrix Gesamt]

Die Hauptmerkmale die besonders herausgestellt wurden, sind die Performance sowie die Dokumentation der Einrichtung und der Verwendung. Fällt viel Netzwerkverkehr an, der aufgezeichnet werden soll, ist es wichtig, die Daten in minimaler Zeit zu verarbeiten ohne einen Teil verwerfen zu müssen. Gleichzeitig muss der Speicherverbrauch sich dabei in einem moderaten Maß bewegen, um andere Dienste nicht auszubremsen.

Der *passiveDNS-client* bot keine geeignete Schnittstelle zur Performancemessung an. Da das Werkzeug von externen Servern abhängig ist, wäre eine derartige Messung zudem nicht aussagekräftig. Dieser Umstand hat keine Auswirkungen auf die Bestwertung, da auch die Maximalpunktzahl in dieser Kategorie zu keiner Dominanz des Werkzeuges geführt hätte. Bei der Dokumentation kam es vor allem auf die Einrichtung und die Verwendung an. Wie bereits im Analyseteil 3.3.1 geschildert, nehmen diese Aspekte einen besonderen Stellenwert ein. Ohne eine entsprechende Anleitung zur Einrichtung könnte man keines der Werkzeuge installieren. Der nächste Schritt ist dann die Bedienung, die genauso gut dokumentiert werden sollte.
Als bestes Werkzeug geht *YAF* mit 57 von 64 Punkten aus der Wertung hervor. Es hat in den stark gewichteten Kategorien die beste Leistung erzielt. Den zweiten Rang nimmt *tshark* mit 51 Punkten ein. Die Performance war hier gegenüber den anderen Werkzeugen signifikant schlechter. Hier muss allerdings gesagt werden, dass *Wireshark* nicht explizit zur Verarbeitung von passiven DNS-Daten optimiert ist, sondern ein allgemeines Netzwerkanalyse Werkzeug mit einer Vielzahl von Funktionen ist.

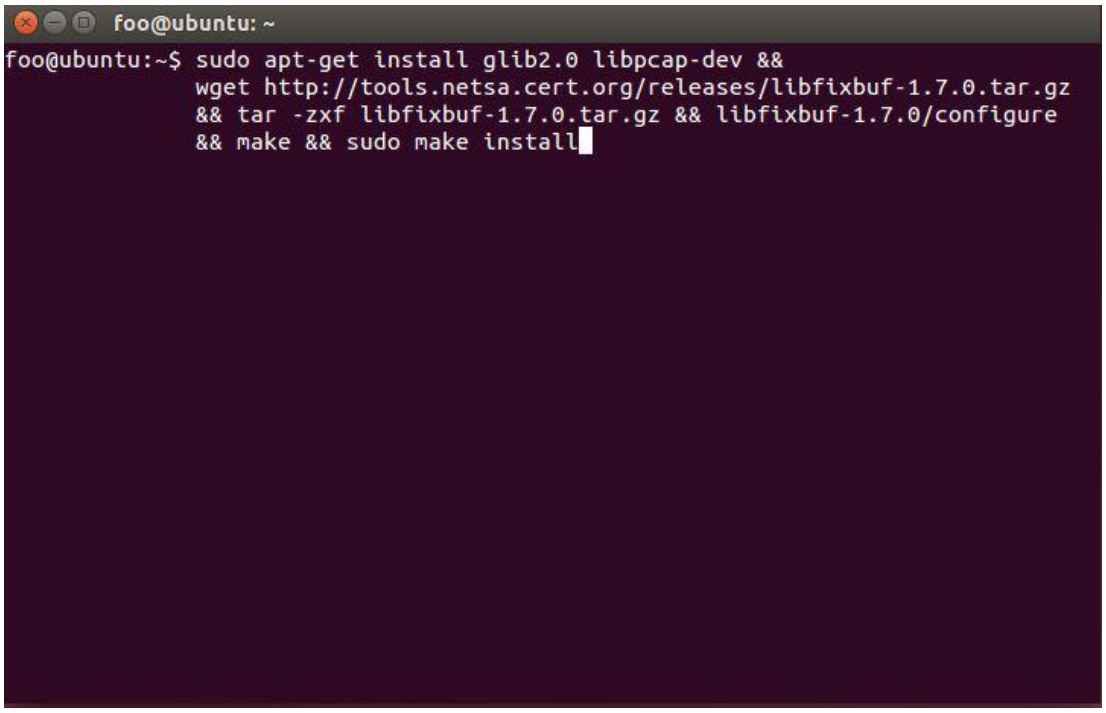
3.7 Ergebnisse

Die Endbewertungen der Werkzeuge liegen teilweise nicht sehr weit auseinander. Einen klaren Gewinner mit großem Abstand gibt es nicht. Jedes Werkzeug hat seine Stärken und Schwächen. Bei PassiveDNS liegen die Stärken in der Verständlichkeit der Dokumentation. Wireshark und der passiveDNS-client erreichten beim Thema Dokumentation sogar in allen Kategorien die Bestnote. Wireshark hat dies der weiten Verbreitung in einer großen Community und einem langen Bestehen zu verdanken. YAF überzeugt beim Punkt Zusatzwerkzeuge. Diese sind aufeinander abgestimmt und bieten einen umfangreichen Funktionsumfang für den Umgang mit DNS-Daten. DNScap ist Spitzenreiter wenn es um die Performance geht.
Kriterien wie die Dokumentation sind fest gegeben und können lediglich von den Autoren beeinflusst werden. Performance Messungen dagegen sind mit Vorsicht zu genießen. Die Laufzeit aller Werkzeuge lag während der Messung im Millisekunden Bereich. Hier können Schwankungen in der Messgenauigkeit auftreten. Aus diesem Grund wurden Durchschnittswerte aus mehreren Programmdurchläufen gebildet. Das Messen in einem solchen Grenzbereich bleibt aber dennoch eine Herausforderung. Viele Prozesse des Betriebssystems, wie *Scheduling*, *Caching* oder das *Memory Management*, können unbemerkt Einfluss auf die Ergebnisse nehmen.

4 Erfahrungsbericht zu YAF

4.1 Einrichtung

Das Repository von *YAF* liegt nicht in einem Paketmanager vor, somit muss das Werkzeug manuell installiert werden. Dies geschieht über *autotools*. Nach vorheriger Konfiguration erzeugt *autotools* ein *makefile* mit dem sich das Werkzeug installieren lässt. Das Installationsarchiv ist auf der Website des CERT NetSA verfügbar. Zusätzlich werden noch die Bibliotheken *glib (*ab Version 2.6.4)*, libcap, sowie libfixbuf* (ab Version 1.0.0) benötigt. Es wird empfohlen zu überprüfen, ob die Bibliotheken bereits auf dem System vorhanden sind. Sie werden häufig bereits mit Linux Distributionen ausgeliefert und sollten idealerweise über die entsprechenden Bordmittel konfiguriert werden.
Auf einer Standardinstallation von Ubuntu 12.04.5 x64 fehlen alle drei Bibliotheken. Mittels des Paketmanagers Aptitude können *glib, sowie libpcap* unter den Paketnamen *glib2.0 und libpcap-dev* installiert werden. Die Bibliothek *libfixbuf* wird vom CERT NetSA bereitgestellt. Diese muss heruntergeladen und kompiliert werden. Die Folgende Darstellung (Dar. 26) zeigt die Installation der Abhängigkeiten von YAF in einer Zeile.



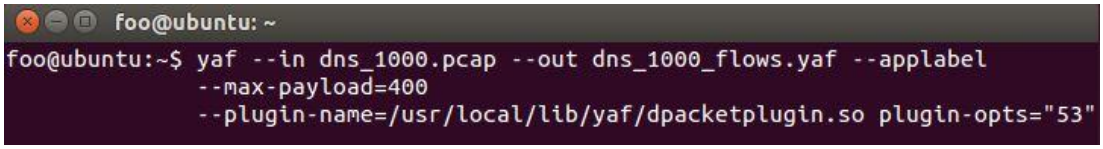
[Dar. 26 Installation Abhängigkeiten YAF]

Nach der Installation der notwendigen Bibliotheken kann nun YAF selbst kompiliert und installiert werden. Bei der Konfiguration des *makefile* werden die Parameter *–enable-applabel* und *–enable-plugins* gesetzt.
Der erste Parameter aktiviert die Funktionalität Protokolle in Netzwerkflussdaten zu erkennen. Wenn ein Protokoll erkannt wird, setzt YAF eine 16-bit Kennzeichnung. Standardmäßig besteht die Kennzeichnung aus dem Port des Protokolls. Bei HTTP-Verbindungen also beispielsweise 80.
Der zweite Parameter sorgt dafür, dass Zusatzfunktionen mit installiert werden. Hierzu gehört die *deep packet inspection*, also eine genauere Analyse der Datenpakete. Hiermit können protokollspezifische Daten erfasst und extrahiert werden.
Um den Export der Daten zu vereinfachen, gibt es Vorlagen, in denen die protokollspezifischen Informationsfelder definiert sind.
Im Anschluss an die Installation müssen die Pfade der Bibliotheken aktualisiert werden. Dies geschieht durch den Befehl *ldconfig*.
YAF für sich kann Netzwerkdaten im *IPFIX* oder pcap Format einlesen. Dabei kann die Quelle der Daten sowohl eine Netzwerkschnittstelle als auch eine Datei sein. Die Ausgabe kann lokal erfolgen, aber auch an externe Prozesse auf entfernten Systemen weitergeleitet werden.

4.2 Verwendung mit super_mediator

Zur weiteren Verarbeitung der Netzwerk Daten wird ein zusätzliches Werkzeug Namens *super_mediator* benötigt. Es implementiert die Funktionalität, IPFIX Daten lesen und schreiben zu können. Gleichzeitig sind diverse Filter enthalten, um relevante Informationen aus dem Netzwerkfluss zu extrahieren. Die Ausgaben können einerseits in Dateien geschrieben, andererseits aber auch in Datenbanken wie MySQL oder PostgreSQL importiert werden. Das Werkzeug wird, genau wie YAF, mittels *autotools* installiert.
Einige Funktionen lassen sich direkt in der Kommandozeile ausführen. Der volle Funktionsumfang lässt sich aber nur mit Hilfe der zugehörigen Konfigurationsdatei nutzen. Hier können sog. *EXPORTER* für verschiedene Datenformate und Protokolle eingerichtet werden. Diese bestimmen die Daten eines Protokolls, dessen Format, sowie den Bestimmungsort.
Standardmäßig ist hier auch eine DNS Deduplikations Funktion definiert. Diese zählt, wie oft ein Domainname aufgerufen wurde und erhöht entsprechend einen Zähler. Zudem wird der Wert des Informationsfeldes *last seen* auf den aktuellen Zeitpunkt angepasst. Somit können DNS Einträge effizienter in einer Zeile archiviert werden.

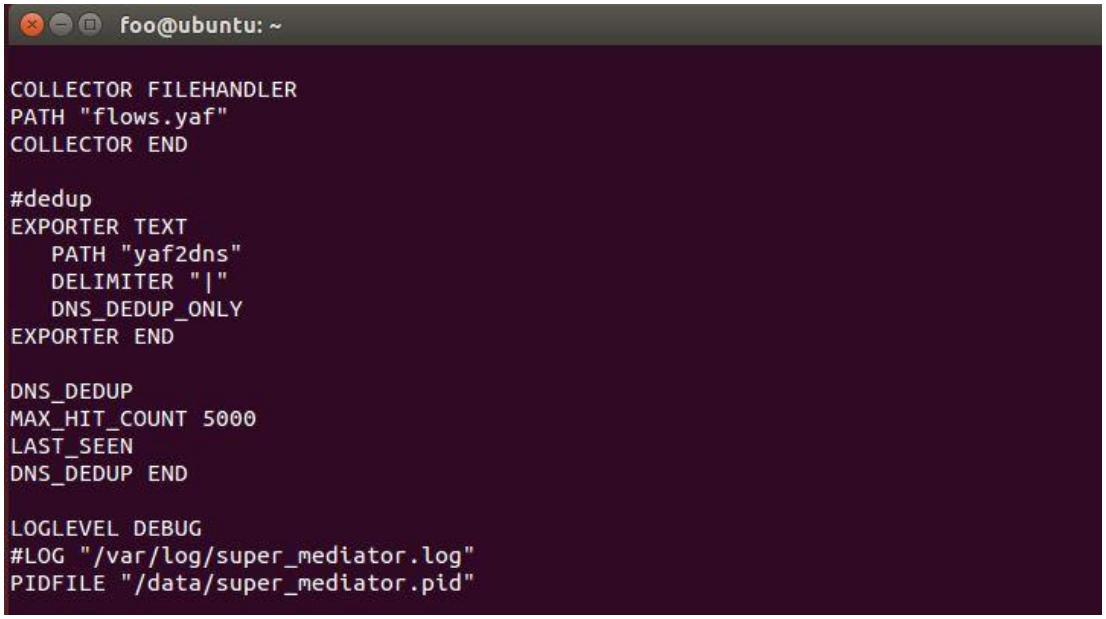
Als Anwendungsbeispiel für die Datenverarbeitung, wird die pcap Datei *DNS_1000.pcap* aus dem Performance Teil dieser Arbeit mit YAF zu einer IPFIX Datei konvertiert.



[Dar. 27 Konvertierung PCAP zu IPFIX]

Zu sehen sind hier (Dar. 27) die Parameter für *deep packet inspection* und *applabel*. Zusätzlich wird DNS-Datenverkehr über den Port 53 gefiltert, um andere Protokolle auszuschließen.
Es wäre auch möglich gewesen, die Flussdaten mittels YAF direkt von der Netzwerkschnittstelle aufzuzeichnen. Dies können auch externe Werkzeuge wie beispielsweise *tcpdump* übernehmen, die häufig schon auf Linux Distributionen vorinstalliert sind.

Die konvertierte Netzwerkfluss Datei kann nun mit Hilfe von super_mediator verarbeitet werden. Hierzu wird zunächst die Konfigurationsdatei angepasst. Diese sieht wie folgt aus (Dar. 28) :

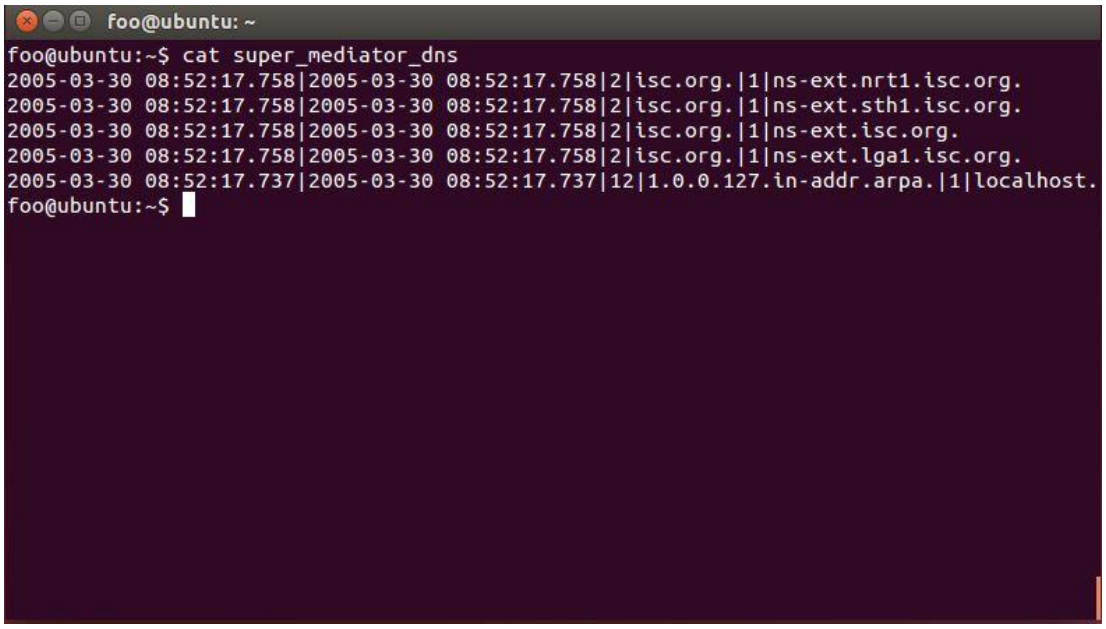


[Dar. 28 Konfigurationsdatei super_mediator]

Die Konfiguration ist in Blöcke unterteilt. Den Anfang bildet der *COLLECTOR* Block. Hier ist die Datenquelle der Netzwerkdaten definiert. Diese kann entweder eine Datei im IPFIX Format sein (*FILEHANDLER*) oder eine TCP Verbindung (*TCP*). Standardmäßig wird bei TCP die Verbindung auf Port 18001 erwartet.
Als nächstes folgen die *EXPORTER*. Es dürfen mehrere *EXPORTER* parallel verwendet werden. Die Datenausgabe hängt vom jeweiligen *EXPORTER* Typ ab. *TEXT* steht für eine

Ausgabe im ASCII Zeichenformat. *FILEHANDLER* hingegen exportiert die reinen Flussdaten in eine Datei. Das Verzeichnis für die zu schreibenden Daten wird mit Hilfe der *PATH* Variable festgelegt und muss vorab manuell angelegt werden.
Verarbeitet werden die Daten die zuvor im COLLECTOR angegeben wurden. Dabei können verschiedene Parameter gesetzt werden.

Beispielsweise steht *DELIMITER* für ein Trennzeichen, welches zwischen einzelnen Datenfeldern eingesetzt wird. Damit lassen sich CSV Dateien mit beliebigem Trennzeichen generieren, um diese in andere Werkzeuge importieren zu können.
Der *DNS_DEDUP* Block enthält die oben genannte DNS Deduplikations-Funktion, sowie Parameter zur Verarbeitung von DNS-Daten. *LAST_SEEN* bewirkt an dieser Stelle eine Ausgabe von bestimmten DNS-Datenfeldern (Dar.29).



[Dar. 29 Ausgabe DNS super_mediator]

Es handelt sich dabei um die Felder:

`first_seen | last_seen | rrtype | rrname | hitcount | rrval`

first_seen enthält das Datum an dem ein Domainname das erste Mal aufgelöst wurde, *last_seen* entsprechend das letzte Datum der Auflösung.
Darauf folgen *rrtype* und *rrname* für den Typ, und den Namen der zugehörigen Domäne des DNS Ressource Records. Der Zähler für die oben genannte DNS Deduplikation wird in *hitcount* hinterlegt. Den Schluss bildet *rrval*, in dem der Inhalt des Ressource Records gespeichert wird. Eine weitere Option für diesen Block ist *FLUSH_TIME* in Sekunden, um

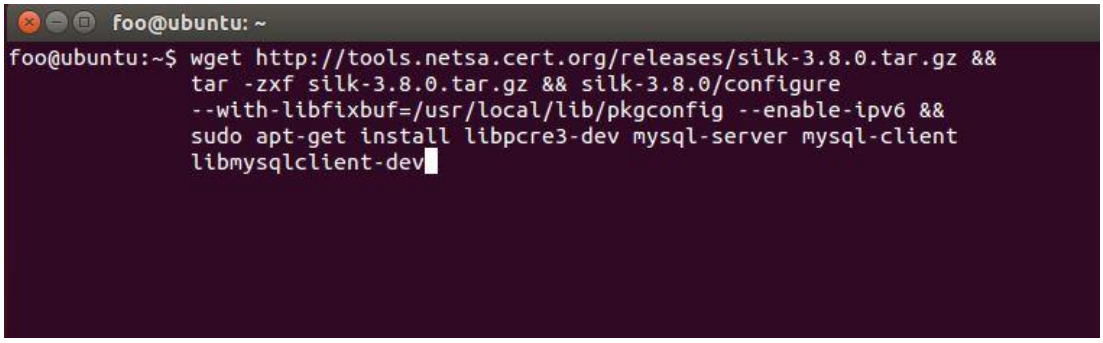
Einträge aus dem Cache zu entfernen, die in der angegebenen Zeit nicht wieder aufgezeichnet wurden. Diese Option sorgt für eine aktuelle Sicht auf die angefragten Domainnamen, indem selten angefragte Domainnamen aus dem Datensatz entfernt werden.

Bei einer forensischen Analyse zur Feststellung einer Infektion durch Schadsoftware, könnten allerdings genau solche verworfenen Datensätze hilfreich sein. Wird ein Domainname innerhalb eines Netzwerks lediglich einmal aufgelöst, ist dies auffällig. Eine Schadsoftware könnte beispielsweise Kontakt zu einem Steuerungsserver aufgenommen haben, um neue Verbindungsdaten oder Anweisungen zu erhalten. Anschließend sammelt die Software unbemerkt Daten und schickt diese über eine neue IP-Adresse an denselben Server. Um solche Fälle frühzeitig erkennen zu können, bietet sich der Abgleich derartiger Einträge mit einer öffentlichen DNS-Datenbank an. Wird zusätzlich zu *FLUSH_TIME* noch die Option *LAST_SEEN* gesetzt, so werden die entsprechenden Einträge nicht verworfen, sondern mit exportiert. Der Parameter *MAX_HIT_COUNT* bietet dieselbe Funktionalität wie *FLUSH_TIME*, allerdings wird hier der Zähler aus der Deduplikationsfunktion als Schwellenwert verwendet.
Durch das aktivierte *deep packet inspection* Plugin ist es auch möglich, auf andere Datenfelder zuzugreifen. Dazu ist ein *DPI_CONFIG* Block in der Konfigurationsdatei notwendig, in dem die gewünschten Felder ausgewählt werden. Ergänzend hierzu kann auch ein *SSL_CONFIG* Block definiert werden, um die Informationen einer SSL Verbindung zu extrahieren. Diese Blöcke werden hier nicht weiter ausgeführt, da die zusätzlichen Informationen nicht das DNS Protokoll betreffen.

4.3 Export in MySQL Datenbank

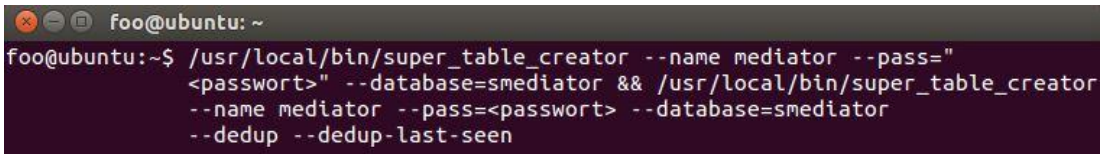
Die Möglichkeiten die gewonnenen Daten zu analysieren erhöhen sich, wenn man diese in eine Datenbank exportiert. Außerdem sind Datenbanken, im Gegensatz zu Dateien, eine effizientere Methode größere Datenaufkommen zu speichern. Es wird allerdings ein gewisses Fachwissen auf dem Gebiet vorausgesetzt, um Datenbankabfragen tätigen zu können.
Um alle Informationen aus den Netzwerkflussdaten entnehmen und in eine Datenbank speichern zu können, gibt es zusätzlich noch eine Sammlung diverser Netzwerk Analyse Werkzeuge vom CERT NETSA. Diese wurden unter dem Namen *SiLK* zusammengefasst und sind auf der Website der Carnegie Mellon University verfügbar.
Weiterhin werden zusätzliche Bibliotheken benötigt. Die in der Dokumentation angegebenen Bezugsquellen sind nicht mehr verfügbar und müssen durch aktuellere Versionen ersetzt werden.

Somit ergibt sich der folgende Installationsbefehl für SiLK, inklusive aller Abhängigkeiten und MySQL Datenbank (Dar.30):



[Dar. 30 Installation SiLK und Abhängigkeiten]

Das Paket *pcre-devel* wird durch libpcre3-dev ersetzt, anstelle von *mysql-devel* werden nun mysql-client und *libmysqlclient-dev* genutzt.
Die Installation von SiLK erfolgt über *autotools*. Anschließend werden laut Dokumentation die Standard Konfigurationsdateien in die entsprechenden Verzeichnisse kopiert und angepasst. Hierunter fällt zum Beispiel die Auswahl des Netzbereiches in dem Datenpakete gesammelt werden sollen. Der nächste Schritt ist die Erstellung der Datenbank samt Benutzer, sowie das Anlegen der Tabellen. Für die Erstellung der Tabellen werden Scripte mitgeliefert. Die Dokumentation verweist auf eine ältere Version dieser Scripte. Dies hat zur Folge, dass der DNS Parameter *–DNS-dedup* nicht mehr funktioniert, da er durch *–dedub* ersetzt wurde. Die berichtigten, vollständigen Befehle zur Erstellung der Tabellen ohne Backslashes lauten also wie folgt (Dar. 31):



[Dar. 31 Anlegen der SQL Tabellen]

Nachdem die Tabellen angelegt wurden, muss ein neuer *EXPORTER* in der *super_median* Konfiguration angelegt bzw. ein bestehender erweitert werden. Die zusätzlichen Angaben umfassen den Namen der Datenbank, die Tabelle sowie die Zugangsdaten. Die ausgegebenen Dateien werden dann geparst und in die Datenbank eingetragen.
Parameter für rotierende Datensätze können ebenfalls gesetzt werden. So kann sichergestellt werden, dass nur Dateien verarbeitet werden, die nicht in Verwendung sind.

```
EXPORTER TEXT
   PATH "/data/smediator/dpi"
   ROTATE 10
   MULTI_FILES
   DPI_ONLY
   LOCK
   MYSQL_USER "mediator"
   MYSQL_PASSWORD "bar"
   MYSQL_DATABASE "smediator"
EXPORTER END
```

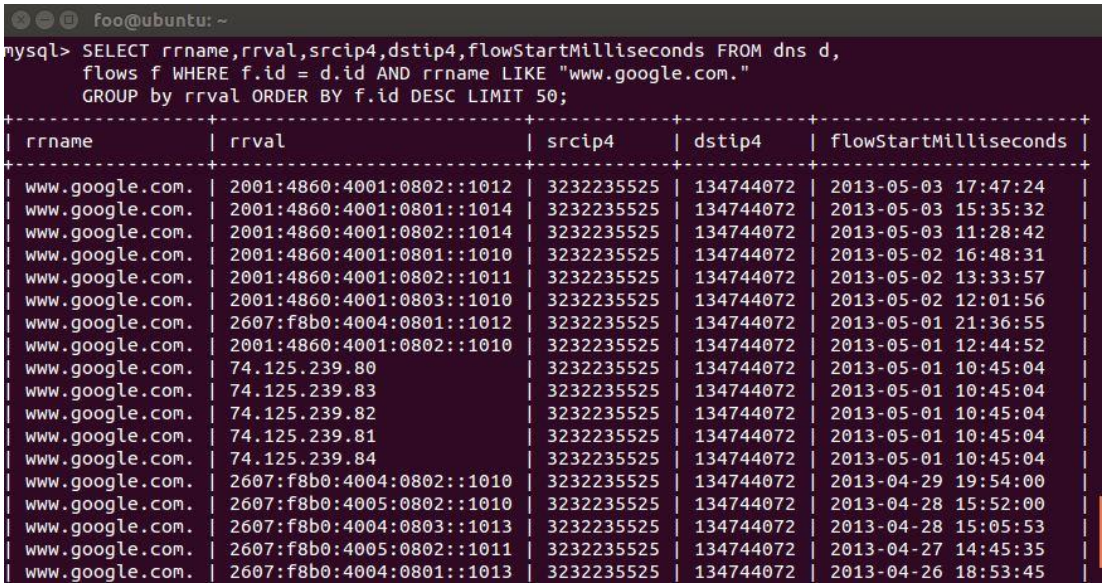
[Dar. 32 MySQL Exporter Konfiguration]

Dar.32 zeigt die verwendete Konfiguration, mit rotierenden Datensätzen im zehn Sekunden Takt. *LOCK* verhindert den Zugriff auf die gerade verwendete Datei für externe Prozesse. Ob die Datenbank Daten enthält lässt sich mit dem folgenden Befehl überprüfen.

```
foo@ubuntu: ~
mysql> select table_name, table_rows from information_schema.tables where table_schema = DATABASE();
+-------------+------------+
| table_name  | table_rows |
+-------------+------------+
| dhcp        |          0 |
| dns         |      73414 |
| flow        |      39946 |
| ftp         |         36 |
| http        |      77462 |
| imap        |         78 |
| irc         |        224 |
| mysql       |          0 |
| nntp        |          0 |
| p0f         |          0 |
| pop3        |         12 |
| rtsp        |          0 |
| sip         |          0 |
| slp         |          0 |
| smtp        |         96 |
| ssh         |         44 |
| tftp        |          0 |
| tls         |      34370 |
+-------------+------------+
```

[Dar. 33 MySQL Datenüberprüfung][36]

In Dar.33 ist ein erfolgreicher Datenimport dargestellt. Zur Verdeutlichung wurde hier ein Datensatz mit unterschiedlichen Protokollen gewählt. Relevant wäre hier lediglich die Tabelle *dns*. Wenn sichergestellt ist, dass die Daten korrekt importiert wurden, lassen sich nun SQL Abfragen formulieren.



foo@ubuntu: ~

```
mysql> SELECT rrname,rrval,srcip4,dstip4,flowStartMilliseconds FROM dns d,
       flows f WHERE f.id = d.id AND rrname LIKE "www.google.com."
       GROUP by rrval ORDER BY f.id DESC LIMIT 50;
+-----------------+--------------------------+------------+-----------+----------------------+
| rrname          | rrval                    | srcip4     | dstip4    | flowStartMilliseconds |
+-----------------+--------------------------+------------+-----------+----------------------+
| www.google.com. | 2001:4860:4001:0802::1012 | 3232235525 | 134744072 | 2013-05-03 17:47:24   |
| www.google.com. | 2001:4860:4001:0801::1014 | 3232235525 | 134744072 | 2013-05-03 15:35:32   |
| www.google.com. | 2001:4860:4001:0802::1014 | 3232235525 | 134744072 | 2013-05-03 11:28:42   |
| www.google.com. | 2001:4860:4001:0801::1010 | 3232235525 | 134744072 | 2013-05-02 16:48:31   |
| www.google.com. | 2001:4860:4001:0802::1011 | 3232235525 | 134744072 | 2013-05-02 13:33:57   |
| www.google.com. | 2001:4860:4001:0803::1010 | 3232235525 | 134744072 | 2013-05-02 12:01:56   |
| www.google.com. | 2607:f8b0:4004:0801::1012 | 3232235525 | 134744072 | 2013-05-01 21:36:55   |
| www.google.com. | 2001:4860:4001:0802::1010 | 3232235525 | 134744072 | 2013-05-01 12:44:52   |
| www.google.com. | 74.125.239.80             | 3232235525 | 134744072 | 2013-05-01 10:45:04   |
| www.google.com. | 74.125.239.83             | 3232235525 | 134744072 | 2013-05-01 10:45:04   |
| www.google.com. | 74.125.239.82             | 3232235525 | 134744072 | 2013-05-01 10:45:04   |
| www.google.com. | 74.125.239.81             | 3232235525 | 134744072 | 2013-05-01 10:45:04   |
| www.google.com. | 74.125.239.84             | 3232235525 | 134744072 | 2013-05-01 10:45:04   |
| www.google.com. | 2607:f8b0:4004:0802::1010 | 3232235525 | 134744072 | 2013-04-29 19:54:00   |
| www.google.com. | 2607:f8b0:4005:0802::1010 | 3232235525 | 134744072 | 2013-04-28 15:52:00   |
| www.google.com. | 2607:f8b0:4004:0803::1013 | 3232235525 | 134744072 | 2013-04-28 15:05:53   |
| www.google.com. | 2607:f8b0:4005:0802::1011 | 3232235525 | 134744072 | 2013-04-27 14:45:35   |
| www.google.com. | 2607:f8b0:4004:0801::1013 | 3232235525 | 134744072 | 2013-04-26 18:53:45   |
```

[Dar. 34 MySQL Datenbankanfrage www.google.de][23]

Dar.34 zeigt einen Ausschnitt einer Datenbankanfrage nach Auflösungen von *www.google.de*. *D*ie Ergebnisse sind auf 50 Einträge limitiert. Dabei können, je nach Anwendung, verschiedene Sortierungen angewandt werden. Es bietet sich beispielsweise an, die Einträge nach dem ersten Erscheinen zu sortieren. So geraten neu aufgetretene Domainnamen schnell in den Fokus. Ist man auf der Suche nach Anzeichen für *Fast Fluxing,* ist es ratsam, die Einträge nach *TTL* oder IP-Adressen zu sortieren.

5 Fazit

5.1 Was wurde erarbeitet?

Das nötige Grundwissen zum Thema DNS wurde ausführlich behandelt. Verschiedene Angriffspunkte der Technik sind näher beleuchtet worden. Dies soll auch ein Bewusstsein für derartige Sicherheitsvorfälle schaffen. Es gibt einige Anzeichen an denen sich ein DNS-Angriff erkennen lässt. Wird das SSL Zertifikat beim nächsten Surfen in einem öffentlichen Netz zur Abwechslung vom Nutzer überprüft, so trägt dieses Bewusstsein bereits Früchte. Wenn jeder nur ein bisschen aufmerksamer wird, schafft das schon ein großes Maß an Sicherheit für alle.
Es wurden Kriterien festgesetzt, um die Güte verschiedener passive DNS Werkzeuge miteinander vergleichbar zu machen. Dabei steht die Wissenschaftlichkeit stets im Vordergrund. Aus diesem Grund sind die Bewertungen möglichst reproduzierbar gestaltet. Subjektive Einschätzungen werden so gut wie möglich begründet, um sie nachvollziehbar zu machen. Hierzu gehören auch diverse Visualisierungen, um den Inhalt besser zu verdeutlichen. Die Bewertungsmatrix die alle Ergebnisse der Analysen zusammenfasst, zeigt auf einen Blick Stärken und Schwächen auf.

5.2 Wurde das Ziel erreicht?

Diese Ausarbeitung hat den Bereich passive DNS grundlegend erläutert. Die Werkzeuge *PassiveDNS, tshark, YAF, Farsight Passive DNS Sensor, DNScap* sowie *passiveDNS-client* wurden objektiv miteinander verglichen und in die Ergebnisse innerhalb einer Bewertungsmatrix visualisiert. Des Weiteren wurde das Werkzeug mit der besten Bewertung ausführlich in Form eines Erfahrungsberichtes beschrieben. Es sollte dem Leser nun möglich sein, die Stärken und Schwächen abzuwiegen und sich für ein *Werkzeug* entscheiden zu können, welches auf seinen Anwendungsfall zutrifft. Somit wurde das gesetzte Ziel dieser Arbeit erreicht.

5.3 Ausblick

Betrachtet man abschließend die Bewertungsmatrix, wird ersichtlich, dass diese noch durch weitere untersuchungswürdige Aspekte ergänzt werden kann. Auch lässt sich die Auswahl der Werkzeuge weiter vergrößern.
Es gibt zahlreiche kleinere Anwendungen mit einem einzigen Anwendungszweck, wie zum Beispiel *Subdomain Bruteforcer,* die auf Basis von Wörterbüchern, die Existenz von Domainnamen verifizieren. Diese könnten zusammengefasst und dokumentiert werden.
Die Bewertungsmatrix selbst lässt sich ebenfalls weiter auszubauen. Hierzu können weitere Kriterien hinzugefügt und mit einer entsprechenden Gewichtung versehen werden.
Außerdem wäre eine Untersuchung von verschiedenen passiveDNS Datenbanken denkbar. Diese wurden im Rahmen dieser Arbeit lediglich angeschnitten. Dabei könnte die Performance der Datenbanken, deren Inhalt, oder weitere Aspekte miteinander verglichen werden. Ein weiteres Feld eröffnet sich durch die Analyse von *Whois* Daten. Dabei könnten statistische Untersuchungen durchgeführt werden, um bestimmte Muster in den Inhaberdaten von schadhaften Domains zu erkennen. Diese Muster ließen sich dann in Werkzeuge integrieren, um die frühzeitige Erkennung verdächtiger Aktivitäten zu verbessern.

Literaturverzeichnis

Informationen:

[1] BSI: *B 5.18 DNS-Server*. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/baust/b05/b05018.html (abgerufen am 26.04.2015).

[2] BSI: *M 2.450 Einführung in DNS-Grundbegriffe*. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02450.html (abgerufen am 26.04.2015).

[3] BSI: *G 5.78 DNS-Spoofing.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05078.html (abgerufen am 12.05.2015).

[4] BSI: *G 5.153 DNS-Amplification Angriff.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05153.html (abgerufen am 03.05.2015).

[5] BSI: *G 5.143 Man-in-the-Middle-Angriff.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05143.html (abgerufen am 10.05.2015).

[6] BSI: *G 5.152 DNS-Hijacking.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05152.html (abgerufen am 10.05.2015).

[7] BSI: *G 5.151 DNS-Flooding.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05151.html (abgerufen am 10.05.2015).

[8] BSI: *G 5.157 Phising und Pharming.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05157.html (abgerufen am 10.05.2015).

[9] BSI: *M 4.353 Einsatz von DNSSEC.* https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04353.html (abgerufen am 10.05.2015).

[10] BSI: *IP-Adresse IPv4, IPv6.* https://www.bsi.bund.de/SharedDocs/Glossareintraege/DE/I/IP-Adresse.html (abgerufen am 10.05.2015).

[11] Christoph H. Hochstätter: *Conficker nicht zu stoppen: Warum er in Unternehmen wütet.* http://www.zdnet.de/41000390/conficker-nicht-zu-stoppen-warum-er-in-unternehmen-wuetet (abgerufen am 21.07.2015).

[12] Cloudflare: *Deep Inside a DNS Amplification DDoS Attack.* https://blog.cloudflare.com/deep-inside-a-DNS-amplification-ddos-attack/ (abgerufen am 10.05.2015).

[13] D-Link: *Sicherheitspatches für D-Link Router DIR-810L, DIR-826L, DIR-626L, DIR-636L.* http://www.dlink.com/de/de/support/support-news/2015/march/05/update-on-router-security (abgerufen am 10.05.2015).

[14] heise Security: *Fast Flux*. http://www.heise.de/security/artikel/Fast-Flux-271526.html (abgerufen am 10.02.2015).

[15] Florian Weimar: *Passive DNS Replication*. http://www.enyo.de/fw/software/dnslogger/first2005-paper.pdf (abgerufen am 10.02.2015).

[16] Free Software Foundation: *GNU General Public License*. http://www.gnu.org/licenses/gpl-3.0.de.html (abgerufen am 10.06.2015).

[17] Microsoft: *Microsoft Security Bulletin MS08-067 – Kritisch*. https://technet.microsoft.com/library/security/ms08-067 (abgerufen am 21.07.2015).

[18] Microsoft: *Resource Record Types*, unter: https://technet.microsoft.com/de-de/library/cc958958.aspx (abgerufen am 10.02.2015).

[19] NIST: *Vulnerability Summary for CVE-2014-0160*. https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0160 (abgerufen am 10.02.2015).

[20] NIST: *Vulnerability Summary for CVE-2014-3566.* https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-3566 (abgerufen am 10.02.2015).

[21] Robert Westervelt: *For Google, DNS log analysis essential in Aurora attack investigation*. http://searchsecurity.techtarget.com/news/1514965/For-Google-DNS-log-analysis-essential-in-Aurora-attack-investigation (abgerufen am 21.07.2015

[22] Robert Westervelt: *Hackers used IE zero-day in Google, Adobe attacks, McAfee says.* http://searchsecurity.techtarget.com/news/1378911/Hackers-used-IE-zero-day-in-Google-Adobe-attacks-McAfee-says (abgerufen am 21.07.2015).

[23] Stephen Reese: *Passive DNS collection and analysis using YaF and Mediator*. https://www.rsreese.com/passive-dns-collection-and-analysis-using-yaf-and-mediator (abgerufen am 21.07.2015).

[24] Wikipedia: *Domain Name System*. https://de.wikipedia.org/wiki/Domain_Name_System (abgerufen am 26.04.2015).

[25] Wikipedia: *Root-Nameserver*. https://de.wikipedia.org/wiki/Root-Nameserver (abgerufen am 26.04.2015).

[26] Wikipedia: *hosts-Datei*. https://de.wikipedia.org/wiki/Hosts-Datei (abgerufen am 26.04.2015).

[27] Wikipedia: *DNSChanger*. https://de.wikipedia.org/wiki/DNSChanger (abgerufen am 26.04.2015).

Passive DNS Werkzeuge:

[28] Chris Lee: *passiveDNS*-client. https://github.com/chrislee35/passiveDNS-client (abgerufen am 10.02.2015).

[29] Edward Fjellskål: *PassiveDNS Replication*. https://github.com/gamelinux/passiveDNS (abgerufen am 10.02.2015).

[30] Farsight Security: *Passive DNS Sensor*. https://www.farsightsecurity.com/Overview/ (abgerufen am 10.02.2015).

[31] Farsight Security: *sie-DNS-sensor*. https://github.com/farsightsec/sie-DNS-sensor (abgerufen am 10.02.2015).

[32] Internet Systems Consortium: *DNSCAP*. https://www.mankier.com/1/dnscap (abgerufen am 10.02.2015).

[33] Verisign, Inc.: *DNScap*. https://github.com/verisign/DNScap (abgerufen am 10.02.2015)

[34] Software Engineering Institute (Carnegie Mellon University): *YAF*. http://Tools.netsa.cert.org/yaf/yaf.html (abgerufen am 10.02.2015).

[35] Software Engineering Institute (Carnegie Mellon University): *yafscii*. http://Tools.netsa.cert.org/yaf/yafscii.html (abgerufen am 10.02.2015).

[36] Software Engineering Institute (Carnegie Mellon University): *super_mediator*. http://Tools.netsa.cert.org/super_mediator/super_mediator.html (abgerufen am 10.02.2015).

[37] Software Engineering Institute (Carnegie Mellon University): *SiLK*. http://Tools.netsa.cert.org/silk/index.html (abgerufen am 10.02.2015).

[38] Wireshark Foundation: *Wireshark*. https://www.wireshark.org/ (abgerufen am 10.02.2015).

[39] Wireshark Foundation: *Display Filter Reference: Domain Name Service*, unter: https://www.wireshark.org/docs/dfref/d/dns.html (abgerufen am 10.02.2015).

Analyse Werkzeuge:

[40] Canonical Group Limited: *Ubuntu*. http://www.ubuntu.com/desktop (abgerufen am 03.02.2015).

[41] Dominik Charousset: *actor-framework*. https://github.com/actor-framework/actor-framework (abgerufen am 03.05.2015).

[42] Gephi Team: *The Open Graph Viz Platfor*. http://gephi.github.io/ (abgerufen am 03.05.2015).

[43] Offensive Security: *Kali Linux*. https://www.kali.org/ (abgerufen am 03.02.2015).

[44] Thomas Williams, Colin Kelley: *Gnuplot*. http://www.gnuplot.info/ (abgerufen am 03.05.2015).

[45] VMware: *VMware Player*, http://www.vmware.com/de/products/player (abgerufen am 03.02.2015).

Darstellungsverzeichnis

[Dar. 1 DNS Resource Record Typen]

[Dar. 2 Hierarchischer DNS Baum BSI]

[Dar. 3 DNS-Anfrage nach isc.org]

[Dar. 4 DNS Log PassiveDNS]

[Dar. 5 DNS Verkehr der Ubuntu Startseite aufgezeichnet mit Passive DNS]

[Dar. 6 Erläuterung Datenfelder PassiveDNS]

[Dar.7 DNS Verkehr tshark nach Filterung]

[Dar. 8 Generierung des Fingerprints beim Farsight Passive DNS Sensor]

[Dar. 9 Aufzeichnung von DNS-Daten bei Aufruf von Google]

[Dar.10 Visualisierte Ausgabe der Anfrage bundestag.de an VirusTotal)

[Dar. 11 Supporttypen]

[Dar. 12 Shellscript Test_Speicher_Laufzeit Testschleife]

[Dar. 13 Auszug aus einzelner Speicher Datei 16 nach Test von PassiveDNS]

[Dar. 14 Shellscript Test_Speicher_Laufzeit Auswertung]

[Dar. 15 Speicherverbrauch PassiveDNS]

[Dar. 16 Speicherverbrauch tshark]

[Dar. 17 Aufruf von tshark]

[Dar. 18 Speicherverbrauch YAF]

[Dar. 19 Aufruf von YAF]

[Dar. 20 Speicherverbrauch Farsight DNS Sensor]

[Dar. 21 Aufruf Farsight DNS Sensor]

[Dar. 22 Speicherverbrauch DNSCAP]

[Dar. 23 Aufruf von DNSCAP]

[Dar. 24 Zusammenfassung Messergebnisse]

[Dar. 25 Bewertungsmatrix Gesamt]

[Dar. 26 Installation Abhängigkeiten YAF]

[Dar. 27 Konvertierung PCAP zu IPFIX]

[Dar. 28 Konfigurationsdatei super_mediator]

[Dar. 29 Ausgabe DNS super_mediator]

[Dar. 30 Installation SiLK und Abhängigkeiten]

[Dar. 31 Anlegen der SQL Tabellen]

[Dar. 32 MySQL Exporter Konfiguration]

[Dar. 33 MySQL Datenüberprüfung]

[Dar. 34 MySQL Datenbankanfrage www.google.de]

Anhang

Shellscript für Performancemessung mit YAF Aufruf

```
01 #!/bin/bash
02 #===================================================================
03 #
04 #      FILE:  Test_Speicher_Laufzeit.sh
05 #
06 #     AUTOR:  Meik Jejkal
07 #   VERSION:  1.0
08 # ERSTELLT:  11.05.2015 - 12:36:50
09 #
10 #    Zusatz: median.py
11 #===================================================================
12
13 n=100 #Iterationen
14 l=0 #Summe Laufzeit
15 s=0 #Summe Speicher
16 e=0 #Summe Speicher Einträge
17 x=0
18
19 #vorherige Ergebnisse bereinigen
20 yes | rm laufzeit
21 yes | rm auswertung
22 yes | rm -r speicher/*
23
24 while (($x < $n ))
25 do
26  #Test
27  sudo ./caf_run_bench 1000 5 0 laufzeit speicher/speicher_$x
/usr/local/bin/yaf --in dns_1000.pcap --out dns.yaf
28  #Caches leeren
29  sudo sync && echo 3 | sudo tee /proc/sys/vm/drop_caches
30  x=$((x+1))
31  echo "Durchlauf $x beendet"
32  if (($x == $n)) #letzte Iteration
33  then
34      #Speicher Dateien zusammenfassen
35      cat speicher/* > speicher/speicher_ges
36      #Daten sortieren
37      sort -n speicher/speicher_ges > speicher/speicher_ges_sort
38      sort -n laufzeit > laufzeit_sortiert
39      #Median für jeden ms Wert berechnen
```

```
40      cat speicher/speicher_ges_sort | python3 mediane.py >
speicher/speicher_median
41      #Werte aufaddieren
42                  l=$(cat laufzeit | paste -sd+ | bc)
43      s=$(cut -d" "-f2 speicher/speicher_median | paste -sd+ | bc)
44      #Anzahl Zeilen bestimmen
45      e=$(wc -l < speicher/speicher_median)
46      #Werte ausgeben
47        echo 'Laufzeit Durchschnitt:' $(($l/$n))  > auswertung
48      ( echo -n 'Laufzeit Minimum: '   ; head -1
laufzeit_sortiert) >> auswertung
49      ( echo -n 'Laufzeit Maximum: '   ; tail -1
laufzeit_sortiert) >> auswertung
50        echo 'Speicher Durchschnitt:' $(($s/$e))  >> auswertung
51      ( echo -n 'Speicher Minimum: '   ; head -1
speicher/speicher_ges_sort) >> auswertung
52      ( echo -n 'Speicher Maximum: '   ; tail -1
speicher/speicher_ges_sort) >> auswertung
53  fi
54 done
```

Python Script zur Berechnung des Median

```
01 #!/usr/bin/python3
02 #===============================================================
03 #
04 #      FILE:  median.py
05 #     AUTOR:  Meik Jejkal
06 #   VERSION:  1.0
07 # ERSTELLT:  11.05.2015 - 12:36:50
08 #
09 # Script für die Berechnung des einfachen Medians
10 #===============================================================
11
12 import sys
13 #Daten aufteilen und einlesen
14 daten = [ line.strip().split(" ") for line in sys.stdin ]
15 p = None
16 werte = None
17 #Berechnungsschleife
18 for t,v in daten:
19     if t != p:
20         if werte:
21             werte.sort()
22             print("%s %s"%(p,werte[(int)(len(werte)/2)]))
23         werte = []
24     werte.append(v)
25     p = t
26 werte.sort()
27 #Werte ausgeben
28 print("%s %s"%(p,werte[(int)(len(werte)/2)]))
```

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____