Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Nick Diedrich

Erkennung von Angriffsmustern in Passiv-DNS-Daten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Nick Diedrich

Erkennung von Angriffsmustern in Passiv-DNS-Daten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 28. April 2017

Nick Diedrich

Thema der Arbeit
Erkennung von Angriffsmustern in Passiv-DNS-Daten

Stichworte
DNS, passiv, DGA, Fast Flux, Phishing

Kurzzusammenfassung
Diese Arbeit erkundet Muster, die sich in vergangenen Attacken auf IT Infrastruktur in passiven DNS Daten gezeigt haben und stellt Methoden vor, um diese zu erkennen.
Nick Diedrich

Title of the paper
Detection of attack pattern in passive DNS data

Keywords
DNS, passiv, DGA, Fast Flux, Phishing

Abstract
This document explores pattern that can be found in past attacks on IT infrastructure in passive DNS data and shows methods to detect such pattern.

Inhaltsverzeichnis

1 Einleitung

1.1 Einleitung

Mit der zunehmenden Verbreitung und Bedeutung von Informationstechnologie (IT) in unserer Gesellschaft häufen sich immer mehr Angriffe auf diese[19]. Viele Methoden zur Abwehr greifen erst, nachdem Schwachstellen erkannt wurden und können bis zu ihrer Erkennung ausgenutzt werden. Eine relativ neue Möglichkeit, um bösartige Aktivitäten z.B von Fast Flux Botnetzen, DGA oder auch Phishing in Netzwerken zu entdecken, bietet Passiv-DNS[28]. Hierzu wird in dem zu überwachenden Netz ein Sensor installiert, der passiv alle DNS Anfragen aufzeichnet. Mit diesen Daten kann im Nachhinein ein Angriff nachvollzogen werden. Dies zeigt sich z.B. im Fall von Conficker[8].
Conficker ändert auf infizierten Systemen DNS Einstellungen. Dies führte dazu, dass DNS Anfragen, die bestimmte Stichworte enthielten ins Leere liefen. Betroffen waren unter anderem Antivirenprogramme. Eine Analyse von DNS-Daten zeigte eine besondere Auffälligkeit bei regelmäßig aufgerufenen Domainnamen. Das Datum des letzten Aufrufs lag vergleichsweise weit zurück. Auf Grund dieser Abweichung in den Passiv-DNS-Daten konnten infizierte Systeme erkannt und bereinigt werden. Conficker setzte zudem auch auf Fast Flux, worauf im Verlauf dieser Arbeit noch detailliert eingegangen wird.
In dieser Arbeit sollen Muster für passive DNS-Daten definiert werden, die in vergangen Attacken gehäuft aufgetreten sind. Für diese Angriffsmuster werden im Anschluss Methoden vorgestellt, die das Erkennen von Angriffen ermöglichen. Da sich diese Methoden in Form von Algorithmen automatisieren lassen, steigt die Wahrscheinlichkeit das Angriffe früh erkannt werden. Dies spart Arbeitszeit, da die Daten erst im Fall einer Auffälligkeit durch die Verantwortlichen eingesehen werden müssen. Insbesondere, wenn für den Betreffenden Datenschutz eine hohe Priorität einnimmt, ist dies von Vorteil. In Folge dessen ergibt sich ein entlastender Effekt auf Systemadministratoren, sofern die Rate der Fehlalarme nicht zu hoch ist. Daher soll auch betrachtet werden, wie zuverlässig die vorgeschlagenen Methoden diese Muster erkennen. Dazu werden, nach Möglichkeit selber Tests durchgeführt, sowie die Erfahrungen anderer vorgestellt.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es das Potential von DNS und im Besonderen Passiv-DNS im Bereich der IT-Sicherheit aufzuzeigen. Angriffe, die sich in DNS-Daten erkennen lassen, sollen vorgestellt werden und im Nachhinein Methoden, mit denen man diese finden kann.
Mit dem Inhalt dieser Arbeit ist es dem Leser / der Leserin möglich, selber Passiv-DNS-Daten zu sammeln und diese anschließend auf Muster, die auf Angriffe hindeuten, zu untersuchen.

1.3 Zielgruppe der Arbeit

Diese Arbeit richtet sich an Personen, die im Bereich IT-Sicherheit tätig sind. Hierzu zählen Netzwerk- und Systemadministratoren/innen die sich im Rahmen ihrer Tätigkeit um die Sicherheit ihrer zu verwaltenden Systeme kümmern. Ihnen soll diese Arbeit Muster aufzeigen die in passiven DNS-Daten auf Angriffe hindeuten. Mit den vorgestellten Methoden zur Detektion dieser Muster sollen sie in der Lage sein diese möglichst zuverlässig zu finden.

1.4 Struktur der Arbeit

Zunächst wird ein Einblick in DNS gegeben und im Besonderen Passiv-DNS. Hier soll aufgezeigt werden welche Daten überhaupt im DNS-Protokoll zu finden sind und welche zusätzlichen Daten Passiv-DNS liefert, die im DNS-Protokoll nicht zu finden sind. Außerdem werden Schwachstellen aufzeigt, die das DNS-Protokoll angreifbar machen.
Anschließend werden verschiedene Methoden vorgestellt, die Autoren von Maleware einsetzen und die mit Passiv-DNS-Daten zu finden sind. Hier soll zunächst ein Verständnis für die Angriffsformen geschaffen werden. Es werden Techniken vorgestellt, die Maleware-Produzenten verwenden um eine Detektion zu vermeiden. Anschließend werden Muster definiert, die sich im Fall eines Angriffs in passiven DNS-Daten finden lassen.
Im folgenden Kapitel werden als erstes verschiedene Werkzeuge vorgestellt, die zum Sammeln von Passiv-DNS-Daten geeignet sind. Anschließend wird auf eine Möglichkeit näher eingegangen, wie man Passiv-DNS-Daten sammeln und verarbeiten kann. In Folge darauf soll für jedes der zuvor definierten Muster ein Algorithmus vorgestellt werden, der eine Detektion ermöglicht. Es wird erläutert, wo Probleme bei den vorgestellten Verfahren liegen und wie hoch die zu erwartende Erfolgsquote ist. Danach wird eine Gesamtarchitektur für das Finden von Angriffsmustern in passiven DNS-Daten aufgezeigt.
Abschließend wird eine kurze Zusammenfassung mit einem Ausblick gegeben. An dieser Stelle wird das erarbeitete Thema resümiert und die Frage, ob das Ziel der Arbeit erreicht

wurde beantwortet. Zudem werden Wege aufgezeigt, wie die erarbeiteten Inhalte weitergehend genutzt und erweitert werden können.

2 Analyse

2.1 DNS

Der Domain Name System (DNS) ist ein global, dezentralisierter Netzdienst, dessen Aufgabe die Beantwortung von Anfragen zur Namensauflösung ist. DNS ist baumartig mit Blättern und Knoten strukturiert, welche als Label bezeichnet werden. Diese Labels können verbunden werden und ergeben dann einen Domain-Namen [10]. 1983 wurde DNS von Paul Mockapetris entworfen[42] und die ersten Spezifikationen in den RFC Standards 882[46] sowie 883[47] durch die Internet Engineering Task Force festgelegt. Diese wurden später von RFC1034[44] und RFC1035[45] abgelöst. DNS ermittelt zu einem Hostnamen die IP-Adresse und umgekehrt. Dies vereinfacht für Menschen die Nutzung von Netzwerken und des Internets, da sich Hostnamen die aus Wörtern bestehen besser als Zeichen- und Zahlenkolonnen in Form von IP-Adressen merken lassen. Im Fall von IPv4 bestehen IP-Adressen aus 4 Blöcken, die durch Punkte getrennt werden und Werte von 0-255 annehmen können. Bei IPv6 handelt es sich um 128 Bit große Adressen, die hexadezimal notiert werden[12]. Eine weit verbreitete Analogie für DNS ist die des Telefonbuches, in dem die Telefonnummern auf Namen abgebildet werden. DNS-Einträge werden als Resource Records (RR) bezeichnet und können von verschiedenen Typen sein [43]2.1. Resource Records werden in Zonendateien eingetragen und verwaltet. Gemäß der Hierarchie werden die Zonendaten der autoritären Server regelmäßig von der darunter liegenden Ebene angefragt. Der SOA Resource Record legt den Zeitpunkt hierfür fest. Um die Version der Zonendatei auf mehreren Servern der gleichen Zone konsistent zu halten, wird die Seriennummer dieses Eintrags benutzt. Die Zonendatei wird übernommen, wenn bei einem Zonenabgleich eine höhere Seriennummer festgestellt wird. Wie oben bereits beschrieben, findet die Namensauflösung hierarchisch in Form eines Baumes statt. Abbildung 2.2 zeigt den schematischen Aufbau der DNS-Hierarchie. Die Baumwurzel ist die höchste Ebene, wird als Punkt dargestellt und Root genannt. Die Top-Level-Domains liegen auf der folgenden Ebene, wie beispielsweise com, org, net. Wenn ein Domainname aufgelöst werden soll, versucht das anfragende System zunächst die zugehörige IP-Adresse im eigenen DNS-Cache zu finden. Falls dies erfolglos ist, wird die systemeigene Hostdatei zur Namensauflösung herangezogen. Ist

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

Abbildung 2.1: DNS Record Typen mit Beschreibungen

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, wird die Anfrage an einen der DNS Root Server weitergeleitet, der die IP-Adresse des zuständigen Nameservers zurück liefert. Hier kann nun die gesuchte IP-Adresse des aufzulösenden Domainnamens angefragt werden. Subdomains werden von diesem Server ebenfalls aufgelöst. Wurde der angefragte Domainname erfolgreich aufgelöst, wird die entsprechende IP-Adresse für eine gewisse Zeit (TTL) im lokalen Cache des anfragenden Systems zur weiteren Verwendung gespeichert [10]. Es existieren 13 DNS Root Server[45]. 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[4]. 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 manipuliert werden müsste[11]. In einem Dokument des IETF aus dem Jahr 2016 wird ein Standard vorgeschlagen, wie DNS über TLS funktionieren kann[30]. Dieser hat den Vorteil, dass die Vertraulichkeit der Daten gegeben ist. Allerdings geben selbst HTTPS Verbindungen das Ziel von Paketen unverschlüsselt im

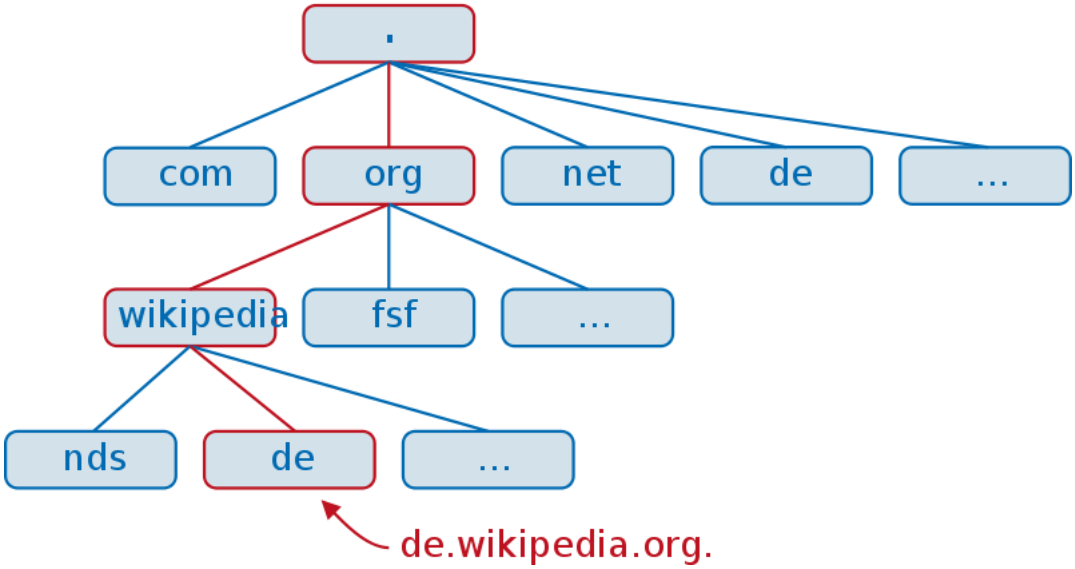

Abbildung 2.2: Schematischer Aufbau der DNS Hierarchie [71]

Header an. 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 [34] 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. Auf Grund dieses Problems wurden Hostdateien 1983 durch DNS abgelöst[42]. Hosts Dateien werden heute lediglich in lokalen bzw. virtuellen Rechnernetzen oder zu Filterzwecken verwendet. Sie sind immer noch vorhanden und dienen, neben dem eigenen DNS-Cache, als erste Quelle der Namensauflösung im System. Durch Angabe des Loopback-Interfaces (127.0.0.1) in der Hostdatei lässt sich beispielsweise erreichen, dass Anfragen an das Loopback-Interface ins Leere laufen, da sie auf das lokale System umgeleitet werden. Unter Linux liegt die entsprechende Datei üblicherweise im Verzeichnis *etc./*.

2.2 Passiv-DNS

Auf Grund der fehlenden Historie ist es nur schwer möglich DNS-Informationen zur Klärung von IT-Sicherheitsvorfällen zu nutzen. Wichtige Informationen fehlen, z.B. wann ein Domainname erstmals in einem Netzwerk aufgelöst wurde. Mittels Reverse Lookups[6] ist es

```
#timestamp||dns-client ||dns-server||RR class||Query||Query Type||Answer||TTL||Count
1322849924.408856||10.1.1.1||8.8.8.8||IN||upload.youtube.com.||A||74.125.43.117||46587||5
1322849924.408857||10.1.1.1||8.8.8.8||IN||upload.youtube.com.||A||74.125.43.116||420509||5
1322849924.408858||10.1.1.1||8.8.8.8||IN||www.adobe.com.||CNAME||www.wip4.adobe.com.||43200||8
1322849924.408859||10.1.1.1||8.8.8.8||IN||www.adobe.com.||A||193.104.215.61||43200||8
1322849924.408860||10.1.1.1||8.8.8.8||IN||i1.ytimg.com.||CNAME||ytimg.l.google.com.||43200||3
1322849924.408861||10.1.1.1||8.8.8.8||IN||clients1.google.com.||A||173.194.32.3||43200||2
```

Abbildung 2.3: Beispiel Ausgabe von einer Passive-DNS Log-Datei[20]

möglich zu einer IP-Adresse den zugehörigen Domainnamen zu ermitteln. Dies basiert auf PTR Resource Records in separaten DNS-Zonen. Da sich diese aber nicht immer automatisch aktualisieren, wenn sich etwas in der Haupt DNS-Zone ändert, ist die Aktualität dieser Daten nicht gewährleistet. Außerdem beschreiben sie nur den Ist-Zustand. Informationen darüber, welche Clients einen Domainnamen aufgelöst haben sind nicht vorhanden. Diese Information kann im Fall einer Infektion durch Schadsoftware nützlich sein, um infizierte Systeme zu identifizieren und zu bereinigen. Auf Grund dieser Gegebenheit wurde Passiv-DNS oder auch Passiv-DNS-Replication 2004 von Florian Weimer entwickelt. Mit dieser Technik ist es möglich opportunistisch einen Teil der Daten, die global im Domain Name System vorhanden sind in einer Datenbank zu indizieren und bei Bedarf zu durchsuchen. Passive DNS-Datenbanken sind sehr nützlich für eine Vielzahl von Aufgaben. Malware und online Kriminalität sind stark von DNS abhängig. So genannte Fast Flux Botnets missbrauchen DNS mit regelmäßigen Updates und niedrigen TTL. Passiv DNS-Datenbanken können helfen Fragen zu beantworten die mit dem Standard DNS-Protokoll nur schwer zu beantworten sind.[60][69]

- Wohin hat die Domain in der Vergangenheit gezeigt?
- Welche Domainnamen werden von welchen Nameserver gehostet?
- Welche Domainnamen zeigen in ein gegebenes IP Netzwerk?
- Welche Subdomains existieren unterhalb einer bestimmten Domain?

Zum Sammeln der Daten werden Sensoren im Netzwerk installiert, welche die gewünschten Daten aufzeichnen. Die aufgezeichneten Daten können dann vor Ort ausgewertet werden oder an eine zentrale Stelle weitergeleitet werden, wo man sie in einer Datenbank indiziert. Wie die Datei für z.B. das Werkzeug “Passiv-DNS“[20] aussieht zeigt Abbildung 2.3. Um den Ursprung verdächtiger Aktivitäten zu ermitteln ist der Zeitstempel von großem Nutzen. Mit ihm kann man z.B. feststellen wie oft sich die IP-Adresse ändert auf die eine Domain zeigt. Dies kann beim Erkennen von Fast Flux Botnets hilfreich sein. Durch den Zähler wird vermieden das gleiche Zeilen mehrfach angezeigt werden müssen, was zu mehr Speichereffizienz führt. Solche Datensätze können dann bei Anbietern wie Farsight[59] hochgeladen werden die verschiedene

Analysen auf passiven DNS-Daten anbieten. Die Sensordaten bieten mehr Informationen als jene, die man aus DNS-Anfragen bekommen kann. Unter anderem gehören dazu Whois-Anfragen um Inhaberdaten zu einer Domain zu ermitteln. Passiv-DNS kann sowohl präventiv als Überwachungsmittel eingesetzt werden, als auch zur Aufklärung von IT-Sicherheitsvorfällen die in der Vergangenheit liegen. Im Fall des Einsatzes zur Prävention werden die Daten automatisch analysiert und bei einem Verdachtsfall ein Systemadministrator informiert. Dieser kann zeitnah entsprechende Maßnahmen ergreifen, bevor potentieller Schaden entsteht.

2.3 Passive DNS-Datenbanken

Passive DNS-Daten nur im eigenen Netzwerk zu sammeln, würde nicht das volle Potential von Passiv-DNS ausnutzen. Um im Nachhinein besser Cyberangriffe aufklären zu können ergibt es Sinn, Sensoren möglichst breit verteilt im Internet und in internen Netzen aufzustellen. Dadurch steigt die Wahrscheinlichkeit, dass für die auf Klärung eines Angriffes relevanten DNS-Daten aufgezeichnet wurden.
Es gibt einige Unternehmen die solche Datenbanken betreiben. Im Folgenden werden einige Datenbanken kurz vorgestellt.

- farsightsecurity.com DNSDB beinhaltet über 13 Milliarden Domainnamen. Zugriff auf die Datenbank ist für kommerzielle Zwecke nur gegen Bezahlung möglich. Farsightsecurity bietet laut eigener Aussage kostenlosen Zugriff auf die Datenbank sowohl für Sicherheitsbehörden als auch für Forscher. Anfragen an die Datenbank können über HTTP und JSON gestellt werden.[33]

- Mnemonic.no hatte im Februar 2015 eine Größe von etwa 1 Milliarden Einträgen. Auch hier gibt es kostenlose und kostenpflichtige Zugriffsmöglichkeiten[3].

- Riskiq.com macht keine Aussagen über die Größe ihrer Datenbank. Zugriff ist über eine API möglich. Dieser Zugriff ist anfangs kostenlos. Folgekosten sind auf den ersten Blick nicht klar zu erkennen. [57]

- TCPIPUtils beinhaltet ca. 250 Millionen Domains. Die Benutzung der API ist kostenpflichtig. Abgerechnet wird pro Anfrage an die Datenbank.[68]

- VirusTotal gibt keine Angaben über die Größe ihrer Datenbank. Das Durchsuchen dieser ist über eine Website kostenlos möglich. Es gibt außerdem eine API die aber nicht für kommerzielle Zwecke genutzt werden darf[67].

- BFK.de macht ebenfalls keine Angaben über die Größe ihrer Datenbank. Nicht automatisierte Suchanfragen können über eine Website gestellt werden. Für Anfragen zu automatisierten Suchanfragen soll man sich bei Interesse mit dem Betreiber der Datenbank in Kontakt setzen.[7]

- CIRCL gibt Freigabe auf seine Datenbank nur nach persönlicher Kontaktaufnahme. Über die Größe der Datenbank werden keine Angaben gemacht. Sie bieten APIs für Python, Ruby und Scala wie auch eine Rest API[15].

2.4 Sicherheit von DNS

Sicherheit hat beim Entwurf von DNS, wie bei der meisten Software, die zu Beginn des Internets entwickelt wurde, keine große Rolle gespielt. Dies hat ihren Ursprung darin, dass die meisten Netzwerke nicht öffentlich waren. Mit dem Wachstum, welches das Internet in den 1990 Jahren, gerade im kommerziellen Bereich erlebte, haben sich diese Anforderungen geändert. Inzwischen sind Maßnahmen zum Schutz und Integrität von Daten, wie auch solche zur Nutzerauthentisierung gefordert[5].
Um DNS diesen neuen Anforderungen gegenüber zu wappnen wurde DNSSEC (Domain Name System Security Extensions)[11] von der Internet Engineering Task Force entwickelt. DNSSEC bezeichnet eine Liste von Spezifikationen um bestimmte Informationen von DNS zu Sichern. Zudem bietet es Origin-Authentication von DNS Daten, Authenticated-Denial-Of-Existence[23] und Datenintegrität, aber keine Vertraulichkeit.
Im Folgenden werden nun verschiedene Angriffe vorgestellt, die auf DNS möglich sind.

2.4.1 DNS-Cache-Poisoning

Beim DNS-Cache-Poisoning, auch DNS-Spoofing genannt, werden kompromittierte Daten in den Cache des DNS-Resolvers eingefügt. Dies sorgt dafür, dass der Nameserver eine falsche IP-Adresse zurück gibt. Ein Angreifer kann also Datenverkehr auf seinen eigenen Computer umleiten. Zum Durchführen dieser Attacke nutzt der Angreifer Schwachstellen in der DNS-Software aus. Ein Server sollte die DNS-Antwort korrekt validieren um sicherzustellen, dass sie von einer autoritativen Stelle stammt. Dieses Validieren kann z.B. mit DNSSEC stattfinden. Wenn der Server dies nicht tut, kann es passieren, dass er inkorrekte Einträge lokal Cached und sie anderen Nutzern weitergibt[52].

2.4.2 DNS-Hijacking

Beim DNS-Hijacking oder auch DNS-Redirection wird so in das Domain Name System eingegriffen, dass eine falsche Antwort auf eine Anfrage gegeben wird. Dies kann durch das Einsetzen von Maleware erreicht werden, die auf dem Computer des Opfers die TCP/IP Konfiguration auf einen DNS-Server ändert, der unter der Kontrolle des Angreifers steht oder durch das Modifizieren eines vertrauenswürdigen DNS-Servers. Ein so modifizierter DNS-Server ist dann nicht mehr Softwarekompatible[9]. Diese Modifikationen können z.B. zum Phishing missbraucht werden. Auch ISP setzen DNS-Hijacking ein, um auf eigene Webserver umzuleiten auf denen sie Werbung verteilen[61], Nutzerstatistiken erheben oder auch um Domains zensieren zu können. Die Liste der ISP, die dies tun ist recht lang und umfasst alle größeren ISPs in den USA. Dies verletzt den RFC Standard für DNS-Antworten[2] und kann Nutzer/innen anfällig für Cross-Site-Scripting machen[61].

2.4.3 DNS-Amplification-Attack

Bei einer DNS-Amplification-Attack handelt es sich um eine Denail-Of-Service-Attack. Bei dieser wird das Domain Name System missbraucht, um große Datenströme auf den Internetanschluss des Opfers umzuleiten. Ziel ist es durch Überlastung den Internetanschluss des Opfers unbrauchbar zu machen[66].
Der Angriff nutzt aus, dass in bestimmten Fällen Nameserver auf kurze Anfragen sehr viel längere Antworten zurückgeben. Der Verstärkungsfaktor kann dabei größer als 50 sein. Zum Umlenken dieser Daten auf das Opfer wird IP-Spoofing benutzt. Im Jahr 2006 wurden 13 DNS-Root-Nameserver das Ziel eines solchen Angriffs. Über 20 Minuten war ein Server nicht mehr erreichbar, die anderen nur sehr langsam[1]. Im Jahr 2013 erreichte ein Angriff gegen die Antispam-Organisation spamhaus.org eine durchschnittliche Traffic-Last von 75 GBit/s. Dabei haben die Angreifer lediglich 750 MBit/s als abgehende Bandbreite benötigt[54].

2.4.4 DNS-Flood

DNS-Flood ist eine Variante von UDP-Flood. Da DNS-Server UDP zur Namensauflösung benutzen, kann das massive Senden von DNS-Anfragen an einen DNS-Server alle seine Ressourcen konsumieren und ihn damit spürbar verlangsamen. Dies führt zu langsameren Antwortzeiten für legitime DNS Anfragen. Während einer DNS-Flood können Server über einen Zeitraum von mehreren Stunden mit Milliarden von DNS Anfragen pro Minute angegriffen werden [75].

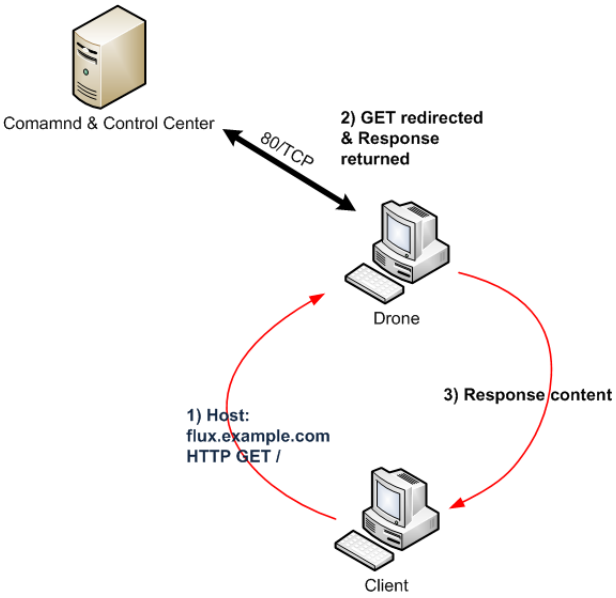

Abbildung 2.4: Fast-Flux Netzwerk[72]

2.5 Angriffe die sich in Passiv-DNS-Daten finden lassen

2.5.1 Fast Flux

Fast Flux[56] ist eine Methode, die unter anderem von Botnets eingesetzt wird. Normalerweise löst ein FQDN (Fully-Qualified-Domain-Name) wie ”example.com” über eine vergleichsweise lange Zeit auf den gleichen Adressraum auf. Eine FQDN mit Fast Flux, die als Kommando und Kontrollserver dient, löst zu einer Vielzahl von IPs auf, zwischen welchen in hoher Frequenz gewechselt wird. Dazu wird eine Kombination aus Round Robin DNS und einer kurzen TTL genutzt. Website-Hosts können dadurch z.B. alle 3 Minuten mit einer neuen Menge von IP-Adressen assoziiert werden. Ein Browser der sich alle 3 Minuten mit einer solchen FQDN verbindet, würde sich so tatsächlich jedes mal mit einem andern infizierten Computer verbinden. Zusätzlich versuchen die Angreifer sicherzustellen, dass die von ihnen genutzten kompromittierten Systeme möglichst viel Bandbreite zur Verfügung stellen und eine hohe Verfügbarkeit aufweisen. Sie benutzen häufig Lastenverteilungsschemata, die den Zustand der verschiedenen infizierten Systeme überwachen. Dadurch wird der zu verbreitende Inhalt möglichst verfügbar gehalten. Eine zweite Schicht, die Blind-Proxy-Redirection nutzt, wird für zusätzliche Sicherheit und Fail-Over verwendet[29]. Diese weitere Schicht erschwert es einzelne Fast-Flux-Netzwerkknoten zu ermitteln und auszuschalten. Der große Pool von rotierenden IP-Adressen ist nun nicht mehr die Endstation für angeforderte Inhalte. Stattdessen dienen die

kompromittierten Front-End-Systeme nun nur noch als Umleiter zu Backend-Servern, die dann Anfragen bedienen. Die Domainnamen und URLs lösen also nicht mehr zu einer IP-Adresse eines bestimmten Servers auf, sondern fluktuieren zwischen vielen Proxies, die selbst die Anfragen an Backend-Server weiterleiten. Fast-Flux-Mutterschiffe sind das kontrollierende Element hinter Fast-Flux-Netzwerken und weisen Ähnlichkeiten zu anderen Kommando und Kontrollsystemen, die man in konventionellen Botnetzen findet, auf. Verglichen mit typischen Botnet-IRC-Servern haben Fast-Flux-Mutterschiffe viel mehr Funktionen. Es ist das Mutterschiff, das durch die Front-End-Fast-Flux-Proxies versteckt wird und den eigentlichen Inhalt an Opfer-Clients bei Anfragen ausliefert. Solche Mutterschiffe waren in der Vergangenheit für lange Zeit in der Lage zu operieren. Sie hosten sowohl DNS als auch HTTP-Dienste mit virtuellen Hostingkonfigurationen die in der Lage sind, Inhalte von tausenden von Domains gleichzeitig auf einem einzigen Host zur Verfügung zu stellen. Fast Flux lässt sich in Single-Flux und Double-Flux einteilen. Der gesamte vorherige Text bezieht sich auf Single-Flux. Double-Flux benutzt noch eine zusätzliche Schicht zum Schutz bei der durchgehend die IP-Adresse des autoritativen Nameservers geändert wird. Dies sorgt für eine Resistenz gegenüber IP basierten Blocklisten, da das Blocken der IP nur in dem kurzen Zeitfenster, während die FQDN die IP auflöst, effektiv ist. Dieses Muster ist in sich selbst nicht bösartig und kann auch für gute Zwecke eingesetzt werden. Eine Domain mit hohen Zugriffszahlen kann auch eine große Anzahl an IPs auflösen. In der Regel lösen gutartige Domains auf einen homogenen IP Raum auf, entweder über Besitz der IP, Adressblock oder Geografie. Bösartige Domains haben eine größere Heterogenität in allen genannten Kriterien. Zusammenfassend ergibt sich ein Muster von Domains, die zu vielen IPs auflösen und diese IPs divers sowohl in Besitzt als auch in Geografie sind.

2.5.2 Domain Generation Algorithms

Domain Generation Algorithm (DGA)[16] Malware nutzt einen Algoritmus, um täglich zufällig tausende von Domains zu erzeugen und verbindet sich mit diesen, um mit einem Controller zu kommunizieren. Botnet-Master registrieren eine kleine Menge dieser Domains täglich, um das Botnet am laufen zu halten, wissend dass die Malware letztendlich versuchen wird eine registrierte Domain aufzulösen. DGAs wurden durch *Conficker.C* und *Conficker.B* populär. Sie generierten zunächst 250 Domainnamen am Tag. *Conficker.C* generierte später sogar 50.000 Domain Namen jeden Tag, von denen aber nur 500 registriert waren. Dies gab einer infizierten Maschine eine 1% Wahrscheinlichkeit täglich aktualisiert zu werden. DGAs werden z.B. auch von Ransomware-Trojanern wie CryptoLocker eingesetzt. In Abbildung 2.5 sieht man Quelltext, den CryptoLocker anfangs zum generieren von Domainnamen verwendete. Wenn man dort als

```
def generate_domain(year, month, day):
    """Generates a domain name for the given date."""
    domain = ""

    for i in range(16):
      year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
      month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
      day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
      domain += chr(((year ^ month ^ day) % 25) + 97)

    return domain
```

Abbildung 2.5: Funktion um einen Domainnamen zu generieren[74]

Eingabe den 7 Januar 2014 wählt, lautet die Ausgabe “intgmxdeadnxuyla“. Am folgenden Tag wäre sie ”axwscwsslmiagfah”. Es gibt allerdings auch DGAs die vermeiden, dass ihre Domains sich zu sehr von nicht DGA-Domains unterscheiden. Beispiele dafür sind Domains, die nur teilweise zufällig sind und aus einer beschränken Menge von Buchstaben und Zahlen generiert werden, oder auch Domains die aus mutierten weit verbreiteten Zeichenketten bestehen.
Ein bekannter und effektiver Weg um DGA-Maleware zu stoppen ist die Vorhersage[73] und Registrierung aller möglicher Domains, bevor der Botnet-Master sie registriert. Dies erfordert Reverse-Engineering von Maleware, was viel Arbeit bedeutet. Glücklicherweise haben algorithmisch erzeugte Domains Eigenschaften, die sich von gutartigen Domains unterscheiden. Gutartige Domains sind in der Regel so gewählt, das Menschen sie sich leicht merken können und beinhalten Wörter aus verschiedenen Sprachen. Zusammenfassend ergibt sich bei DGAs ein Muster von Domains, mit anomaler lexikographischer Struktur.

2.5.3 Phishing

Beim Phishing wird versucht sensible Informationen wie Nutzernamen, Passwörter und Kreditkarteninformationen (indirekt auch Geld) zu erbeuten. Dies findet über elektronische Kommunikationswege statt, wobei falsche Identitäten genutzt werden die das Vertrauen des Opfers genießen. [63]. Phishing ist ein Beispiel von Social-Engineering Techniken die genutzt werden um Nutzer/innen zu täuschen und Schwachstellen in der IT-Sicherheit auszunutzen[62].
Beim Spear Phishing werden gezielt Einzelpersonen oder Unternehmen angegriffen. Die Angreifer sammeln Informationen über ihr Ziel, um ihre Erfolgswahrscheinlichkeit zu erhöhen. Diese Angriffe zählen zu den Erfolgreichsten unter allen Phishing Methoden[64].
Beim Clone Phishing wird eine legitime und zuvor versandte E-Mail, die einen Anhang hat

- steamcommuity.ga
- steamecommunity.com
- steamcommunitey.com
- banking.commerzbank.de-vorgangsnummer-aewpkx9t.top
- amazon.co.uk.security-check.ga
- paypalsecure-2016.sucurecode524154241.arita.ac.tz

Abbildung 2.6: Phishing URLs[17]

nur minimal verändert. Der Anhang wird durch eine bösartige Version ersetzt und die E-Mail von einer Spoofed E-Mailadresse versendet, die aussieht, als ob sie vom originalen Absender stammt[18].
Beim Whaling werden gezielt Führungskräfte in Unternehmen angegriffen[24]. Der Inhalt der Mail ist zugeschnitten auf ihren Empfänger, seiner Aufgabe im Unternehmen und beschäftigt sich oft mit Exekutivausgaben, Beschwerden von Kunden oder Gerichtsvorladungen. Die meisten Phishing Methoden setzen auf technische Täuschungsmethoden um URLs legitim erscheinen zu lassen. Eine Möglichkeit hierzu ist eine leicht anders geschriebene Domain wie bei den ersten 3 URLs in Abbildung 2.6. Ansonsten werden auch gerne Subdomains zur Täuschung verwendet. Beispiele hierzu sieht man in Abbildung 2.6 unter den letzten 3 Punkten. Die eigentliche Domain ist dann z.b “de-vorgangsnummer-aewpkx9t.top“ mit den Subdomains ”commerzbank“ und ”banking”. So wird das Opfer dazu gebracht zu denken, es wäre auf “banking.commerzbank.de“[13]. Ein weiteres Problem sind Internationalized Domain Names (IDN). Sie erlauben es in einer Domain anstatt eines lateinischen O ein kyrillisches O zu verwenden. Für ein Opfer würden so zwei unterschiedliche Domains komplett identisch aussehen. Webbrowser setzen inzwischen Algorithmen ein, um gegen diese Problematik vorzugehen [55][48]. In E-Mails und sozialen Netzwerken stellen sie aber immer noch ein Problem dar. Da Phisher möglichst nicht von Filtern erkannt werden wollen, haben einige sogar begonnen Bilder anstatt Texte in Mails zu verwenden. Dies hat zu einer Evolution von Phishing Filtern geführt, die jetzt auch Optical-Character-Recognition und Intelligent-Word-Recognition benutzen[49]. Phishing Angriffe haben in der Vergangenheit oft auf kleine Schreibfehler in der Domain gesetzt oder Markennamen eingesetzt um vertrauenswürdig zu wirken. Im ersten Fall sind die Phishing-Domains leicht veränderte Varianten von echten Domains, sodass sie immer noch ähnlich aussehen. Somit ergibt sich ein Muster von DNS-Abfragen für Domains, die eine

leicht veränderte Version von beliebten Domains sind. In anderen Fällen enthält eine Phishing Domain vertraute Markennamen um legitim zu erscheinen.

2.5.4 Ausreißer

In Statistiken ist ein Ausreißer ein Messwert, der nicht in eine Messreihe passt oder nicht den Erwartungen entspricht[26]. Anomalien in Netzwerken zu finden ist ein wichtiger Bestandteil von Intrusion-Detection-Systems (IDS).[76] Zunächst wählt man eine oder mehrere Eigenschaften des DNS-Verkehrs. Danach wird die normale Spanne, in der die gewählten Eigenschaften auftreten ermittelt. Zuletzt sucht man nach Aufzeichnungen, in denen diese Eigenschaften von dem zuvor ermittelten Normal abweichen. Es folgen einige Beispiele für solche Ausreißer die in passiven DNS-Daten zu finden sein können.

Mail Spam

Eine Möglichkeit um festzustellen, ob aus dem eigenen Netzwerk Mail Spam verschickt wird, ist der Mail Exchanger Record (MX Record). Wenn eine E-Mail versendet wird ermittelt der sendende MTA (Message Transfer Agent) über DNS den MX Record für jede der Domains, an die eine E-Mail geschickt werden soll. Daher hinterlässt das Senden von E-Mails Spuren in Passiv-DNS-Daten. Wenn nun plötzlich eine sehr viel höhere MX Rate als sonst vorliegt, könnte dies auf eine Maleware Infektion hindeuten die Spam verschickt.

NXDOMAIN

DGA-Domains enthalten manchmal englische Wörter, um DGA-Klassifizierer die lexikographische Eigenschaften zur Detektion benutzen, zu täuschen. Ein Beispiel hierfür ist Nivdort[50], dessen Domainnamen immer aus zwei Strings eines Array bestanden. Eine mögliche, von Nivdort generierte Domain wäre somit "withinreport.net". Allerdings hinterlassen DGA-Maleware Spuren, die schwerer zu verschleiern sind. Da die Maleware tausende von Domains registriert und nur einige von ihnen zu Hosts aufgelöst werden können, gibt ein Großteil der DNS-Abfragen einen Code=3 zurück, also eine Non-Existent-Domain oder NXDOMAIN. Normalerweise entstehen NXDOMAINs mit einer Wahrscheinlichkeit von weniger als 5% durch Tippfehler oder Kopierfehler. Auf Maschinen, die mit einer DGA Maleware infiziert sind steigt diese Wahrscheinlichkeit. Charakteristisch für eine Infektion mit DGA-Maleware ist also eine erhöte Rate von NXDOMAIN-Fehlern.[41]

Tägliche DGA-Domains

Für die meisten Domains sieht man tagsüber mehr Aufrufe als nachts und werktags mehr als am Wochenende. Im Fall von Maleware die DGA benutzt sieht man ein anderes Muster. Abgesehen von den schon oben genannten NXDOMAIN werden einige wenige Domains über den Zeitraum bis neue Domains generiert werden überdurchschnittlich viel und gleichmäßig aufgerufen. Nach einer vom Angreifer bestimmten Zeit stehen neue Domains zur Verfügung. Die zuvor stark aufgerufenen Domains fallen komplett weg und werden durch die neuen abgelöst.[41]

2.6 Potential von Passiv-DNS im Bereich der IT-Sicherheit

In der IT-Sicherheit ergeben sich durch Passiv-DNS Möglichkeiten ein Gesamtschutzkonzept zu erweitern. Durch das Vorhandensein einer Historie von DNS-Daten im eigenen Netzwerk wird das Aufklären von Sicherheitsvorfällen erleichtert. Es kann so besser nachvollzogen werden, wie sich ein Angriff abgespielt hat und von wo er ausgegangen ist. Aus der Historie von DNS-Daten lässt sich ein Normalzustand des Netzwerks in dem Sensoren aufgestellt sind erkennen und Veränderungen von diesem werden sichtbar. So lassen sich auch Infektionen mit Schadsoftware erkennen, die Antivirenprogramme noch nicht detektieren. Durch das Verwenden von Passiv-DNS-Datenbanken ist das Detektieren von DNS-Cache-Poisoning nahezu in Echtzeit möglich. Passiv-DNS-Datenbanken wie DNSDB von Farsight Security bieten im 15 Minuten Takt eine Liste von neu gesehenen Domains. Da eine hohe Korrelation zwischen neuen Domainnamen und bösartigen Aktivitäten besteht, können Administratoren diese zunächst präventiv blockieren [38]. Außerdem wäre es möglich, dass bei einem Aufruf eines problematischen DNS-Namen die Anfrage auf eine Website mit Warnhinweisen oder auch auf einen Honeypot umgeleitet wird. Passiv-DNS kann so dazu beitragen technischen Schaden und damit auch potenziell wirschaftlichen Schaden zu verhindern bevor er entsteht.
Hierzu muss lediglich von der für das Netzwerk verantwortlichen Person ein Sensor im zu überwachenden Netzwerk installiert werden.
Konkrete Vorteile für Organisationen, Unternehmen und Hochschuhlen sind vielfach. Die DNS-Daten geben lediglich an welche Domain wann aufgerufen wurde. So kann man trotz Überwachung des Netzwerks sicherstellen, dass der Inhalt der Kommunikation zwischen Client und Server vertraulich bleibt. Ein kompromittieren von HTTPS, wie es teilweise Antivirenprogramme tun ist nicht nötig. Es reicht wenn ein Sensor pro Netzwerk installiert wird. Dieser kann dann passiv den DNS-Verkehr aller im Netzwerk befindlichen Systeme aufzeichnen. Die

anschließende Auswertung der aufgezeichneten Daten kann zu einem späteren Zeitpunkt, an dem die Auslastung des zu überwachenden Systems möglichst gering ist erfolgen.

3 Synthese

3.1 Daten Sammeln

Zum Sammeln von passiven DNS-Daten gibt es eine Vielzahl von Werkzeuge wie Passive-DNS[20], Wireshark[22], Tshark[21], YAF[65], Farsight-DNS-Sensor[58]und DNSCAP[39]. Der Funktionsumfang dieser unterscheidet sich sehr. Es folgt hier eine kurze Beschreibung mit markanten Merkmalen, die bei der Auswahl eines Werkzeugs helfen.

3.1.1 Farsight

Der Farsight-DNS-Sensor lädt aufgezeichnete Daten auf das Farsight-DNSDB-System hoch. Der Client ist quelloffen und die Daten werden anonymisiert. Der passive DNS-Sensor sammelt DNS-Daten, die er von einem Caching-Server erhält. Anfragen von individuellen Clients werden nicht gespeichert. Es wird außerdem die Möglichkeit geboten die IP-Adresse des Auflösers durch Nullen zu ersetzen. Als positiver Nebeneffekt der Nutzung des Sensors verbessert man das Farsight DNSDB System. Der Sensor ist auf Debian, Redhat und FreeBSD verfügbar.

3.1.2 Wireshark / Tshark

Tshark (Konsolenversion von Wireshark) und Wireshark haben einen größeren Funktionsumfang als nur das Aufzeichnen von DNS-Daten, können aber sehr wohl dafür verwendet werden. Es werden hunderte von Protokollen unterstützt, die man mit Wireshark aufnehmen und anschließend analysieren kann. Tshark bietet außerdem die Möglichkeit Datenpakete als JSON zu exportieren und sogar in einer Form, die sich direkt in Elasticsearch einlesen lässt. Es werden Windows, Linux, Mac OS, Solaris, FreeBSD, NetBSD und viele weitere Plattformen unterstützt.

3.1.3 YAF

Wenn man zum indizieren der Daten gerne MySQL oder PostgreSQL benutzen möchte kommt das Tool YAF in Frage. Dieses bietet eine Exportfunktion sowohl für verschiedene Dateiformate

wie auch für MySQL und PostgreSQL. YAF ist ein Netzwerkflussanalysewerkzeug das vom Ceert NETSA entwickelt wurde.

3.1.4 Passive DNS

Passive-DNS kann DNS-Verkehr entweder direkt von der Netzwerkschnittstelle aufzeichnen oder schon aufgezeichnete pcap Dateien einlesen. Die relevanten Daten werden zur Verarbeitung in ein eigenes Dateiformat konvertiert. Es gibt allerdings auch die Möglichkeit, Daten mit Hilfe eines Skripts in einer MySQL Datenbank zu indizieren. Mit diesem passiv DNS Client können Abfragen zu Domainnamen und IP-Adressen an verschiedene passiv DNS-Datenbanken gestellt werden. Unter anderem Passiv-DNS, Mnemonic, DNSDB und CIRCL. Außerdem ist ein Export als visuelles Graphenformat möglich oder auch als CSV, XML, YAML, JSON oder ASCII.

3.1.5 DNSCAP

DNSCAP von Verisign kann DNS-Netzwerkverkehr im pcap Format aufzeichnen. Es unterstützt sowohl IPv4 als auch IPv6. Zudem besitzt DNSCAP eine Zeitsteuerung, um automatisiert Daten in einem festgelegten Zeitfenster aufzuzeichnen.

3.2 Daten verarbeiten und speichern

3.2.1 Elasticsearch

Elasticsearch[14] ist eine Suchmaschine die auf Lucene aufbaut. Sie bietet eine verteilte Textsuchmaschine mit einer HTTP Web-Schnittstelle und JSON schemafreien Dokumenten. Elasticsearch wird in Java entwickelt und der Quellcode unter den Bedingungen der Apache Lizenz veröffentlicht. Ein Einsatz von Tshark zusammen mit Elasticsearch ist gut möglich, da DNS Daten die im pcap Format vorliegen mit Hilfe von Tshark sehr leicht in Elasticsearch indiziert werden können. Eine Automatisierung des Importprozesses ist mit Tshark gut möglich, da es sich im eine Konsolenanwendung handelt. Zuerst wandelt man mit Tshark die aufgezeichneten pcap Dateien in JSON Dateien um, die Elasticsearch einlesen kann. Hierfür bietet Tshark einen eigenen Parameter ”-T ek”. Der Wert “-e“ gibt an, welche Informationen exportiert werden sollen. Ein Beispiel für eine Konvertierung von pcap nach JSON ist in Abbildung 3.1 zu sehen. Mit einem kurzen Bash-Skript ist es so möglich massenhaft pcap Dateien in ein für Elasticsearch lesbares JSON umwandeln. Tshark gibt allen exportierten JSON Dateien den gleichen Index. Es gibt keine Option den Index bei der Konvertierung manuell auszuwählen. Dies lässt sich nach dem Export mit Hilfe eines kurzen Skriptes sehr einfach beheben [Abbildung3.2]. Das

```
for f in $1
do
OUT=$(echo $f| rev | cut -c 5- | rev | echo "$(cat_-)json")
tshark -T ek -e dns.qry.name -e dns.flags.rcode -x -r > $OUT
done
```

Abbildung 3.1: Pcap zu JSON mit Tshark

```
counter=$3
for f in $1
do
sed -i -- 's/'$2'/packets'$counter'/g' "$f"
((counter++))
done
```

Abbildung 3.2: Indices der JSON Dateien ändern.

indizieren in Elasticsearch ist ähnlich simpel, wie man in Abbildung 3.3 sieht. Elasticsearch bietet eine Bulk-API, die viele Indizier oder Löschoperationen in einem einzigen API-Aufruf ermöglicht. Dies erhöht als positiven Nebeneffekt auch die Geschwindigkeit, mit der indiziert wird. Zum Indizieren wird in diesem Fall cURL genutzt. cURL ermöglicht das Senden und Empfangen von Dateien mit einer URL Syntax. Hier wird auf eine lokale Elasticsearch Instanz zugegriffen, die unter dem Port 9200 erreichbar ist. Es gibt aber auch Dienstleister die das Hosten übernehmen. Zwar wird in diesem Beispiel eine HTTP Verbindung genutzt, Elasticsearch beherrscht aber auch HTTPS. Dies wird gerade dann nötig, wenn Elasticsearch auf einem entfernten Server betrieben wird. Das *$f* ist in diesem Fall jeweils die JSON Datei, die eingelesen werden soll. Elasticsearch bietet außerdem die Möglichkeit Netzwerkverkehr mit Beats aufzuzeichnen oder Dateien mit File Beats einzulesen. Beats ist eine leichtgewichtige Plattform um Daten zu sammeln und diese an Logstash und Elasticsearch weiterzuleiten. Diese können dann mit Hilfe von Logstash gefiltert werden, bevor man sie in Elasticsearch indiziert. Daten, die in Elasticsearch indiziert sind, können außerdem mit wenig Aufwand mithilfe von

```
for f in $1
do
  curl -XPUT 'http://localhost:9200/_bulk' --data-binary "@$f"
done
```

Abbildung 3.3: JSON in Elasticsearch indizieren

Kibana visualisiert werden. Elasticsearch bietet offizielle Client APIs für Java, Java REST, Java Script, Groovy, .NET, PHP, Perl, Python, Ruby und noch einige Weitere, die von Nutzern entwickelt werden.
In Abbildung 3.4 werden Beispiele für die Verwendung der Java-API gezeigt. Elasticsearch speichert beim Indizieren nicht nur Werte die indiziert werden, sondern auch das Dokument selbst. Mit der in Abbildung 3.4 implementierten Methode *getValue* kann man so auch direkt in Dokumenten suchen. Der von Elasticsearch zurückgegebene Wert muss dann nur noch von Klammern bereinigt werden. Diese Herangehensweise kann nützlich sein, wenn Elasticsearch Teile des Dokuments nicht indiziert, weil sie zu lang sind.
Ein Beispiel für das Suchen nach indizierten Werten sieht man in Abbildung 3.4 an der Methode *getAllIDs*. Hier wird für alle Einträge vom Typ *pcap_file* der Index mit der dazugehörigen ID ermittelt.

```
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import org.elasticsearch.action.get.GetResponse;
import org.elasticsearch.action.search.SearchResponse;
import org.elasticsearch.client.transport.TransportClient;
import org.elasticsearch.common.settings.Settings;
import org.elasticsearch.common.transport.InetSocketTransportAddress;
import org.elasticsearch.search.SearchHit;
import org.elasticsearch.transport.client.PreBuiltTransportClient;
public class ElasticData {
public ArrayList<String[]> getAllIDs() {
        ArrayList<String[]> al = new ArrayList<String[]>();
        TransportClient client = getClient();
        SearchResponse response = client.prepareSearch()
        .setTypes("pcap_file").setSize(10000).setFetchSource(false).get();
        for (SearchHit hit : response.getHits().getHits()) {
            al.add(new String[] { hit.getIndex(), hit.id() });
        }
        client.close();
        return al;
}
private String getValue
        (String index, String id, String value, TransportClient c) {
        TransportClient client = c;
        try {
                GetResponse response = client
                .prepareGet(index, "pcap_file", id)erden
                .setFetchSource("*" + value, "").get();
                String out = response
                .getSource().values().toArray()[0].toString();
                String regex = ".*\\{" + value + "?=\\[?(.*?)\\]?\\}";
                Pattern p = Pattern.compile(regex);
                Matcher m = p.matcher(out);
                m.find();
                return m.group(1);
        } catch (Exception e) {return "";}
}
private TransportClient getClient() {
        TransportClient client = new PreBuiltTransportClient(Settings.EMPTY);
        try {
                client.addTransportAddress
                (new InetSocketTransportAddress(
                InetAddress.getByName("localhost"), 9300));
        } catch (UnknownHostException e) {
                e.printStackTrace();
        }
        return client;
}
}
```

Abbildung 3.4: Elasticsearch mit Java API

3.3 Mustererkennung

3.3.1 Fast Flux

Das zuverlässige Erkennen von Fast Flux ist nicht einfach. Eine mögliche Methode ist, die Anzahl von A Records für eine FQDN zu überwachen und bei überdurchschnittlich vielen IPs pro FQDN Alarm zu schlagen. Eine etwas komplexere Herangehensweise ist zusätzlich zu der Anzahl der IPs die zu einer Domain gehören auch noch zu ermitteln, auf wie viele Länder diese IPs verteilt sind und wer Besitzer des IP-Raums ist. Andere hatten mit dieser Methode in der Vergangenheit Erfolg und haben Domains gefunden die sich dann auch als bösartig herausgestellt haben[32].

Domain	**IPs**	**Country Codes**	**Besitzer des IP Raums**
ahmdallame.no-ip.biz	34	iq,fr	dynamic ip pool,earthlink ltd. Communications & internet Service
liiion999.zapto.org	45	fr, ma, it, us, hu, at, ro, mx	edis infrastructure in france, mexico server, telentia enterprise customer, amplusnet srl, micfo llc., serverastra kft, india server, dynamic ip pool, adsl maroc telecom, psinet inc, national computer systems co
liiion777.zapto.org	50	fr, ma, us, hu, at, nl, ro, mx	dynamic ip pool, mexico server, maroctelecomasdll, edis infrastructure in spain, telentia enterprise customer, amplusnet srl, serverastra kft., india server, leaseweb netherlands b.v., adsl maroc telecom, psinet inc.

3.3.2 Domain Generation Algorithm

Bei DGAs ergibt das Suchen nach lexikographischen Unterschieden zu normalen Domains Sinn. Um diese Unterschiede erkennen zu können, bietet sich der Einsatz eines Random Decision Forest an[31]. Random Decision Forest wurde von Tin Kam Ho 1995 entwickelt[27].Die Random

Forests sind eine Kombination aus unbeschnittenen Klassifikations- und Regressionsbäumen. Ein Random Forest Algorithmus generiert viele Klassifikationsbäume. Jeder Baum wird mit Hilfe eines Klassifikationsalgoritmus aus verschiedenen Proben der originalen Daten aufgebaut. Nachdem der Wald geformt ist, wird ein neues Objekt, das klassifiziert werden soll jedem der Bäume zur Klassifizierung übergeben. Jeder Baum gibt eine Stimme ab, die angibt welche Klasse das Objekt hat. Der Wald wählt die Klasse, für die das Objekt die meisten Stimmen bekommen hat[76]. In diesem Fall benutzt man Klassifikationsbäume für Ratio von Konsonanten zu Vokalen, die längste Konsonantenfolge, N-Gramm aus Wörterbüchern und Alexa Top Domains. Mit so einem Random Forest ließen sich Domains, die von der Maleware TinyBanker generiert wurden mit einer Genauigkeit von 98% detektieren [32].

Domain von Tiny Banker	**Urteil**	**Sicherheit**
sdprjrntgvlw.ru	DGA	0.98
fnetiyouqksr.xyz	DGA	0.96
cpowrnbskkxt.xyz	DGA	0.99
pmiioppkqrvw.pw	DGA	0.98
brstpvrtkcpp.com	DGA	0.97
htschinwcghk.com	DGA	0.86

Um zuverlässiger sagen zu können, wie gut die Methode wirklich funktioniert, müsste man den Random Forest gegen andere DGA testen. Außerdem könnte man den Random Forest um weitere Sprachen erweitern. In diesem Fall wurde nur auf Eigenschaften der englischen Sprache getestet.

3.3.3 Phishing

Wie schon zuvor beschrieben gibt es beim Phishing mindestens zwei verschiedene Muster, die man finden kann. Zum Einen leicht veränderte Domains, zum Anderen Subdomains die den Nutzer/innen glauben lassen, sie befänden sich auf einer anderen Website. Im Folgenden werden zwei Ansätze vorgestellt, die diese Muster zuverlässig erkennen können.

Search Substring

Bei Phishing Domains die Markennamen einsetzten hilft eine einfache suche nach diesen Namen, um fündig zu werden. In Java lässt sich eine solche Methode in wenigen Zeilen implementieren 3.5. Die Liste von Marken kann man je nach Bedarf erstellen. Es wäre möglich nur auf Marken zu testen, von denen man ausgeht, dass sie im überwachten Netzwerk Ziel von Angriffen werden. Auch wäre es möglich eine Liste der meist gephishten Websites zu benutzen.

```
public ArrayList<String[]> s(ArrayList<String>domain,ArrayList<String>brand){
        ArrayList<String[]> output = new ArrayList<String[]>();
        domain.stream().parallel().forEach(j -> {
                brand.stream().parallel().forEach(i -> {
                        if (j.contains(i) == true) {
                                output.add(new String[] { j, i });
                        }
                });
        });
return output;
}
```

Abbildung 3.5: Methode um Substrings zu suchen in Domainnamen

Diese Methode wurde gegen eine Liste von einigen tausend bekannten Phishing-Domains getestet. Dabei ist es sinnvoll vor dem Suchen - und . Zeichen aus den Domains zu entfernen, da einige Domains Markennamen mit . und - unterteilen. In Tests gegen eine Mischung aus bekannten Phishing Domains und zufällig ausgewählten *Alexa.com* Domains traten keine Probleme mit Fehlerkennungen auf. Allerdings wurden logischerweise Domains mit leicht veränderten Markennamen nicht erkannt.

Levenshtein-Distanz

Bei einer nur leicht veränderten Domain wie z.b. "steamcomunity.com" anstatt "steamcommunity.com" kann man, um ein Maß der Ähnlichkeit zu erhalten, die Levenshtein Distanz verwenden[37]. Die Levenshtein Distanz gibt die minimale Anzahl von Einfüge-, Lösch- und Ersetzoperationen an, um aus einer Zeichenkette eine Andere zu machen. Mathematisch ist die Levenshtein Distanz eine Metrik auf dem Raum der Symbolsequenzen. Sie wurde 1965 von Wladimir Lewenstein entwickelt. Es folgt als Beispiel die Levenshtein-Distanz zwischen "Tier" und "Tor".

1. Tier
2. Toer (i durch o ersetzen)
3. Tor (e löschen)

Die Levenshtein-Distanz ist in diesem Fall zwei. Man kann die Levenshtein Distanz mit folgenden Matrix-Rekurrenzen ausdrücken 3.6. Hierbei stehen u und v für die beiden eingegebenen Zeichketten. Nachdem die Matrix D berechnet ist, steht die Levenshtein Distanz in der Matrix-Zelle $D_{m,n}$. In den Zellen $D_{i,j}$ stehen die Levenshtein Distanzen für $u_{0,i}$ und $v_{0,j}$. Es gibt

$$
\begin{aligned}
m &= |u| \\
n &= |v| \\
D_{0,0} &= 0 \\
D_{i,0} &= i, 1 \leq i \leq m \\
D_{0,j} &= j, 1 \leq j \leq n \\
D_{i,j} &= \min \begin{cases} D_{i-1,j-1} & +0 \; falls \; u_i = v_j \\ D_{i-1,j-1} & +1 \; (Ersetzung) \\ D_{i,j-1} & +1 \; (Einfügung) \\ D_{i-1,j} & +1 \; (Löschung) \end{cases}, \leq i \leq m, 1 \leq j \leq
\end{aligned}
$$

Abbildung 3.6: Levenshtein Distanz[37]

verschiedene Möglichkeiten die Liste der Domains für einen Test nach der Levenshtein Distanz aufzubauen.

- Eine Liste der derzeit global am meisten durch Phishing gefährdetsten Domains wie z.B. Paypal, Facebook, HSBC Group und World of Warcraft [35].
- Eine vom Nutzer nach eigenen Präferenzen erstellte Liste.
- Eine Liste von Domains die in Passiv-DNS-Daten des eigenen Netzwerks regelmäßig auftreten.

Je länger die Liste desto mehr Rechenleistung benötigt der Test. Daher muss eine Balance zwischen Abdeckung und Rechenaufwand gefunden werden. Am Ende muss bei dieser Vorgehensweise immer noch ein Mensch eingreifen. Um die Belastung für diesen so gering wie möglich zu halten, ist es wichtig, dass dieser nicht zu viele Fehlalarme erhält. Hierzu muss man herausfinden ab welcher Levenshtein Distanz es sich lohnt Alarm zu schlagen. Probleme bei dieser Methode ergeben sich bei Domains, die nur eine geringe Levenshtein Distanz voneinander aufweisen, obwohl sie keine Phishing-Domains sind. In Abbildung 3.7 sieht man eine Implementation der Levenshtein Distanz in Java.

3.3.4 Ausreißer

Für das Finden von Ausreißern gibt es eine Vielzahl von Ansätzen. Wenn man von einer Normalverteilung der Daten ausgeht, kommen Verfahren wie Grubbs Test oder auch Mahalanobis Distanz in Frage.

```java
import java.util.ArrayList;
import java.util.stream.IntStream;
public class LevenshteinDistance {
private int minimum(int a, int b, int c) {
        return IntStream.of(a, b, c).min().getAsInt();
}
private int levenshteinDistance(String a, String b) {
        int matrix[][];
        int lenghtA;
        int lenghtB;
        int iterateA;
        int iterateB;
        char ithCharA;
        char ithCharB;
        int cost;
        lenghtA = a.length();
        lenghtB = b.length();
        if (lenghtA == 0) {return lenghtB;}
        if (lenghtB == 0) {return lenghtA;}
        matrix = new int[lenghtA + 1][lenghtB + 1];
        IntStream.rangeClosed(0, lenghtA).forEach(x -> matrix[x][0] = x);
        IntStream.rangeClosed(0, lenghtB).forEach(x -> matrix[0][x] = x);
        for (iterateA = 1; iterateA <= lenghtA; iterateA++) {
                ithCharA = a.charAt(iterateA - 1);
                for (iterateB = 1; iterateB <= lenghtB; iterateB++) {
                        ithCharB = b.charAt(iterateB - 1);
                        if (ithCharA == ithCharB) {
                                cost = 0;
                        } else {
                                cost = 1;
                        }
                        matrix[iterateA][iterateB] = minimum(
                        matrix[iterateA - 1][iterateB] + 1,
                        matrix[iterateA][iterateB - 1] + 1,
                        matrix[iterateA - 1][iterateB - 1] + cost);
                }
        }
        return matrix[lenghtA][lenghtB];
}
public EditDistance distance
(ArrayList<String> strings1, ArrayList<String> strings2, int maxDistance) {
        EditDistance e = new EditDistance();
        strings1.stream().parallel().forEach(i -> {
                strings2.stream().parallel().forEach(j -> {
                                int dis = levenshteinDistance(i, j);
                                if (dis < maxDistance) {
                                        e.add(i, j, dis);
                                }
                });
        });
        return e;
}
}
```

Abbildung 3.7: Levenshtein Distanz in Java

$$G = \frac{\max_{i=1,\dots,N} \left| Y_i - \bar{Y} \right|}{s} \tag{3.1}$$

$$G > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}} \tag{3.2}$$

Abbildung 3.8: Grubbs Test[25]

Grubbs Test

Frank E. Grubbs publizierte 1950 den Test, der auch als "Maximum normed residual test" oder "extreme studentized deviate test" bekannt ist[25]. Grubbs Test basiert auf der Annahme von Normalität. Daher muss man zuerst verifizieren, dass die Daten, auf denen man Grubbs Test anwenden will von einer Normalverteilung abgeschätzt werden können. Der Test entfernt bei jedem Durchlauf einen Ausreißer. Nachdem der Ausreißer entfernt wurde wird der Test wieder angewandt bis kein Ausreißer mehr gefunden wird. Dabei hat jede Iteration eine andere Wahrscheinlichkeit einen Ausreißer zu finden. Der Test ist nicht für Datenmengen die kleiner als sechs sind geeignet, da ansonsten die meisten Datenpunkte als Ausreißer identifiziert werden[51]. In Abbildung 3.8(3.1) sieht man eine Definition von Grubbs Test. Grubbs Test ist die größte absolute Abweichung vom Durchschnitt der Probe in Einheiten der Standardabweichung. Der abgebildete Test zeigt die zweiseitige Version. Es gibt aber auch einseitige Versionen, die testen ob ein Minimalwert oder Maximalwert ein Ausreißer ist.
Eine Implementation in Java sieht man in Abbildung 3.9. GetGrubbs steht hier für den Teil links vom ">" in der Ungleichung [Abbildung3.8(3.2)] und getGrubbsCompareValue für den rechten Teil. Dieser Test eignet sich für das Suchen von Ausreißern im Record Type. Man kann die Anzahl von verschieden Records über einen Tag oder eine Woche summieren und diese anschließend Grubbs Test unterziehen. Das Gleiche gilt für NXDOMAINs. Auch hier müsste man NXDOMAINs eines Tages summieren und dann eine Folge dieser Werte auf Ausreißer testen. In eigenen Tests mit selbst generierten Testdaten hat dies funktioniert. Um eine Aussage über die Effektivität unter Realbedingungen treffen zu können, würde es sich anbieten Tests auf Daten einer passiv DNS Datenbank durchzuführen. Wenn Ausreißer in mehr als einer Dimension gefunden werden sollen, müsste man ein anderes Verfahren wie die Mahalanobis Distanz[40] anwenden.

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.atomic.AtomicReference;
import org.apache.commons.math3.distribution.TDistribution;
import org.apache.commons.math3.stat.StatUtils;

public class GrubbsTest {
public Double getOutlier(ArrayList<Double> data, double significanceLevel) {
        AtomicReference<Double> outlier = new AtomicReference<Double>();
        double size = data.size();
        if (size < 3) {
                return null;
        }
        if (getGrubbs(data, outlier) > getGrubbsCompareValue
                        (data, significanceLevel, size)) {
                return data.get();
        } else {
                return null;
        }
}
public double getGrubbsCompareValue
(ArrayList<Double> values, double significanceLevel, double size) {
        TDistribution tDistribution = new TDistribution(size - 2.0);
        double criticalValue = tDistribution.inverseCumulativeProbability
                        ((1.0 - significanceLevel) / (2.0 * size));
        double criticalValueSquare = criticalValue * criticalValue;
        return ((size - 1) / Math.sqrt(size)) * Math.sqrt
        ((criticalValueSquare) / (size - 2.0 + criticalValueSquare));
}
public double getGrubbs
(ArrayList<Double> values, AtomicReference<Double> outlier) {
        double[] array = toArray(values);
        double mean = StatUtils.mean(array);
        double standardDeviation = standardDeviation(values);
        double maximalDeviation = 0;
        for (Double value : values) {
                if (Math.abs(mean - value) > maximalDeviation) {
                        maximalDeviation = Math.abs(mean - value);
                        outlier.set(value);
                }
        }
        return maximalDeviation / standardDeviation;
}
public Double standardDeviation(ArrayList<Double> values) {
        return standardDeviation(toArray(values));
}
public Double standardDeviation(double[] values) {
        return Math.sqrt(StatUtils.variance(values));
}
public double[] toArray(ArrayList<Double> values) {
        return values.stream().mapToDouble(d -> d).toArray();
}
}
```

Abbildung 3.9: Grubbs Test in Java

4 Gesamtarchitektur

Das folgende Kapitel gibt eine Übersicht, wie eine Architektur für das Sammeln und Auswerten von Passiv-DNS-Daten aussehen kann. Es wird auf die einzelnen Komponenten eingegangen, wie sie zu benutzen sind und wo. So sollen Leser/innen in die Lage versetzt werden die in dieser Arbeit vorgestellen Methoden und Werkzeuge selbst einzusetzen. In Abbildung 4.1 sieht man das Zusammenspiel aller Komponenten. Die Komponenten DNS-Sensor, Filter, Index, Datenbank, Auswertung und Report sind alle Modular. Jede von ihnen kann auf einem eigenen Computer laufen. Der passive DNS-Sensor, in diesem Fall Wireshark zeichnet die DNS-Daten in Form von pcap (packet capture) Datein auf. Diese werden dann wiederum von Wireshark eingelesen und in JSON (JavaScript Object Notation) umgewandelt. Danach werden die JSON Datein mit scp (Secure Copy) verschlüsselt auf den Server übertragen, auf dem die Datenbank läuft. Auf dem Datenbankserver indiziert Curl anschließend die JSON Datein in Elasticsearch. Diese sind dann, wenn Elasticsearch auf dem gleichen System wie die Auswertungskomponente läuft über http://localhost:9200 abrufbar. Für den Fall, dass die Auswertungskomponente auf einem anderen System laufen soll kann Elasticsearch auch von außen zugreifbar gemacht werden. Diese Verbindung nutzt dann auch https und kein http um die Vertraulichkeit und Integrität der Daten sicher zu stellen. Die Auswertungskomponente ist in diesem Fall ein Java Programm. Es verschaft sich Zugriff auf die Daten über die Java-API von Elasticsearch. Wenn eine der Komponenten zum finden von Angriffen einen Fund hat, gibt sie diesen weiter an eine Logkomponente, welche ihn in einer Logdatei vermerkt. Diese Logdatei wird dann, sofern sie nicht leer ist, in einem definierten Intervall mit Hilfe von Mutt per E-Mail an ausgewählte E-Mail-Adressen geschickt. Hierzu wird die Logdatei vor dem Verschicken mit PGP (Pretty Good Privacy) verschlüsselt.

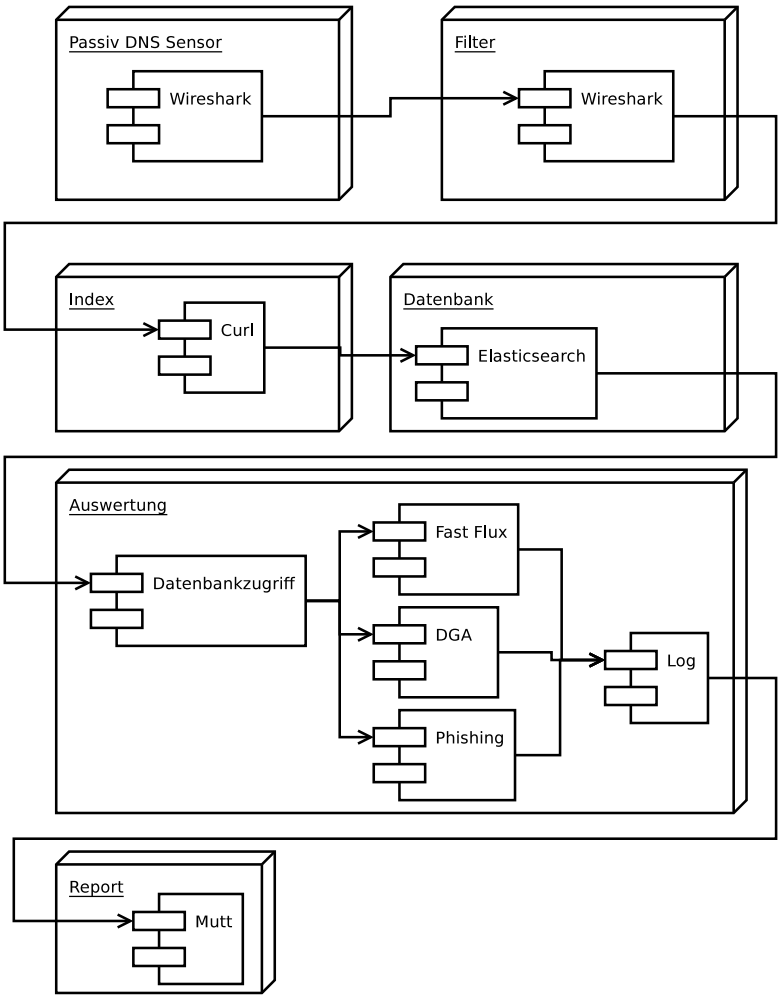

Abbildung 4.1: Architektur des Sensors

4.1 Netzwerkplan

In Abbildung 4.2 sieht man den Plan eines Netzwerks in dem das hier beschriebene System zum Einsatz kommen kann. Die in der Architektur beschriebenen Komponenten würden sich, abgesehen vom Sensor im internen Netz für Server- und Datenressourcen befinden. Ein DNS-Sensor ist in diesem Fall unter anderem mit dem DMZ-Switch verbunden.

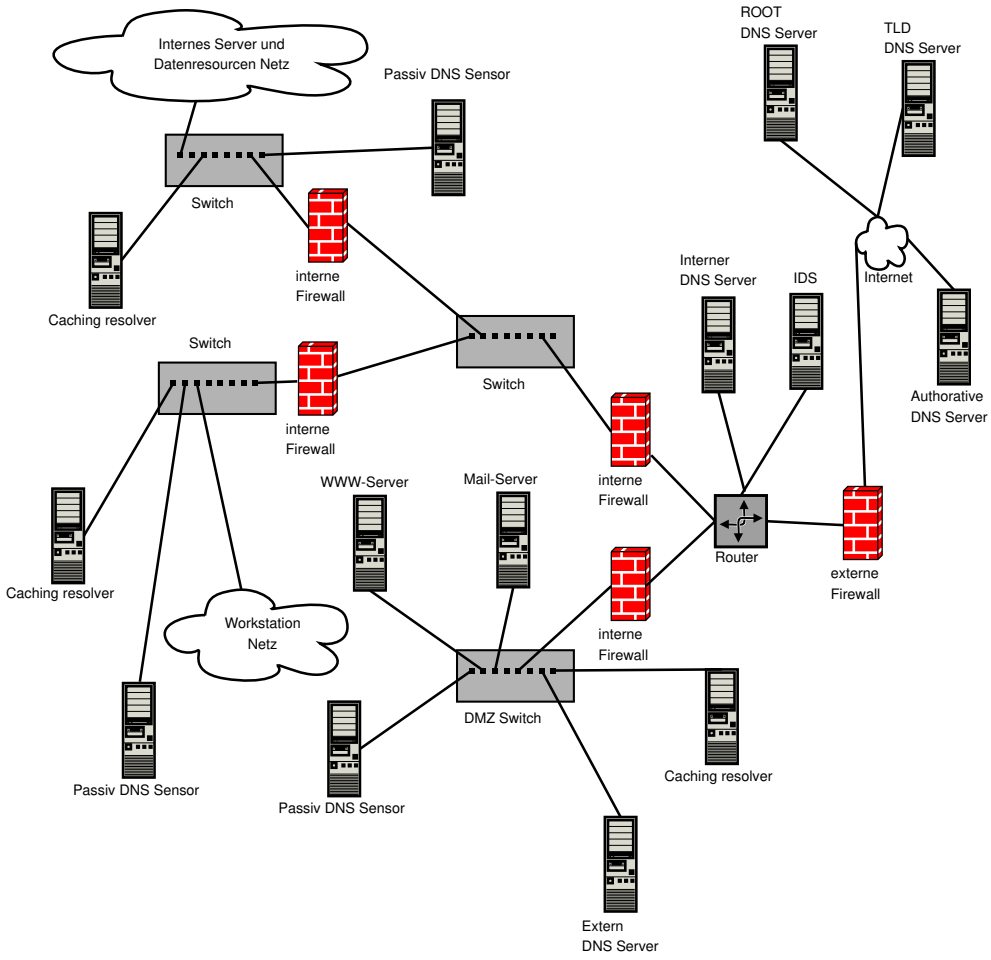

Abbildung 4.2: Netzwerkplan

Sensoren sollten in der Nähe von großen Caching-Nameservern oder an Uplinks von Netzwerken, die Caching-Nameserver enthalten installiert werden. Ein Caching-Nameserver fordert diversere Daten von anderen Nameservern an als ein einzelner autorativer Nameserver. Außerdem macht es Sinn den Sensor so zu positionieren, dass er nicht den Datenverkehr zwischen dem Caching-Resolver und seinen Clients aufzeichnet. Der Sensor kann auf dem Monitor-Port des DMZ-Switches liegen wo er den Datenverkehr des Caching-Nameservers überwacht. In so einem Aufbau ist man nicht an den DNS-Antworten, die an das LAN geschickt werden interessiert, da die Resource-Records schon beim ersten Cachen gespeichert wurden.[70]
In dem in Abbildung 4.2 gezeigten Netz befindet sich je ein Sensor am Switch der drei Subnets. An den verschiedenen Sensoren werden Anfragen der Clients an den jeweiligen Caching-Resolver aufgezeichnet. Dadurch erhält man Informationen wie oft Clients bestimmte Domains

anfragen. Der Passiv-DNS-Sensor am DMZ-Switch würde so Angriffe auf den WWW-Server und Mail-Server aufzeichnen. Hier wäre dann z.b. das Missbrauchen des Mailservers sichtbar um Spam zu verschicken. Aber auch wenn einer der beiden Server von DGA oder Fast Flux Malware befallen wäre. Gleiches gilt für die anderen beiden Subnets für Workstations und für das interner Server. Beim Erkennen eines Angriffs kann dann mit dem hinzufügen von Blockregeln für die Firewalls reagiert werden und unter Umständen einer Veränderung der IDS-Filter.

4.2 Zusammenwirken mit anderen Sicherheitskomponenten

4.2.1 Firewall

Die durch Passiv-DNS ermittelten bösartigen Domains können verwendet werden, um Blockierregeln für Firewalls zu erstellen. Außerdem kann durch das Blockieren von DNS-Anfragen mit IP-Adressen, die von außerhalb des eigenen Nummernraums kommen, das Missbrauchen des eigenen Nameresolvers als Open-Reflector in DDoS Attacken verhindert werden[53].

4.2.2 Virtual Private Network

Virtual Private Networks (VPN) stellen ein Problem dar. Sie tunneln DNS-Anfragen verschlüsselt durch das Netzwerk an den dort installierten passiven DNS-Sensoren vorbei. Daher würde es sich anbieten Methoden zu erkunden um VPNs zu unterbinden.

4.2.3 Wireless Local Area Network

Wenn sich ein System mit einem WLAN verbindet, das nicht zum überwachten Netzwerk gehört bedeutet dies auch ein Umgehen vom passiven DNS-Sensoren und Intrusion Detection Systems. Daher sollte sichergestellt werden, dass sich Nutzer/innen nicht mit beliebigen WLANs verbinden können.

4.2.4 Intrusion Detection System

Intrusion Detection Systems können zusammen mit Passiv-DNS genutzt werden. Als bösartig erkannte Domains können mit Hilfe von IDS wie Snort, Suricata oder auch OSSEC erkannt und blockiert werden. IDS bieten selbst Möglichkeiten Auffälligkeiten im DNS-Verkehr zu erkennen.

4.2.5 Antivirenfilter

Antivirenprogramme wie Kaspersky nutzen Network Driver Interface Specification Filter um Netzwerkpakete abzufangen. Diese Pakete werden dann an Komponenten wie Mail-Anti-Virus, Web Anti-Virus und Anti-Spam weitergegeben[36]. Hier kann eine Überschneidung von Funktionalitäten bestehen.

4.3 Passiv DNS Sensor

In jedem der Netzwerke die überwacht werden sollen wird Tshark als passiver DNS-Sensor installiert. Dieser zeichnet zunächst alle Daten auf Port 53 auf. Damit werden alle DNS-Daten aufgezeichnet, abgesehen von solchen die über Port 853 versendet werden. Port 853 ist für das Benutzen von DNS über TLS reserviert. DNS-Anfragen von Nutzern die das Tor Browser Bundel benutzen werden ebenfalls nicht aufgezeichnet, da diese Anfragen zunächst durch das Tornetzwerk gehen und damit außer Reichweite eines lokalen Sensors liegen.

4.4 Filtern und Konvertieren

Die gespeicherten Daten müssen vor dem indizieren in Elasticsearch noch in ein passendes JSON Format umgewandelt werden. Hierbei macht es auch Sinn alle Daten die später nicht für eine Analyse gebraucht werden zu verwerfen. Zu den Daten die zur Analyse gebraucht werden zählt der Zeitstempel des Packets, die Domain die aufgelöst wurde, Fehlercodes, die IP zur Domain und der Record Typ. Für die Erkennung von Fastflux ist die Information zu welchem Land eine IP gehört nützlich. Hierzu kann man die DNS Daten um diese Informationen erweitern.

4.5 Index

Das Indizieren der von Tshark erzeugten JSON in der Datenbank wird mit Curl vorgenommen. Curl wird per cron Job automatisch alle 24 Stunden ausgeführt, nachdem Tshark mit dem Export der JSON Datein fertig ist.

4.6 Datenbank

Sobald die Daten in Elasticsearch indiziert sind, lassen sie sich wie im Kapitel "Daten verarbeiten und speichern"gezeigt wurde, leicht über die Java API abrufen.

4.7 Auswertung

Die Auswertung der Daten durch die Komponenten findet automatisch statt. Nutzer/innen bekommen dann Informationen, wenn eine der auswertenden Methoden einen potentiellen Angriff gefunden hat. Angemessene Maßnahmen zu ergreifen liegt dann in der Verantwortung des Nutzers.
Für jede der in dieser Arbeit vorgestellten Attacken, gibt es je eine Komponente die für Auswertung der gesammelten Daten zuständig ist. Eine Auswertung von Daten findet einmal am Tag statt. Dies bietet den Vorteil das der teilweise aufwändige Prozess des Erkennens von Angriffen zu Zeiten stattfindet wenn Ressourcen frei sind.

4.7.1 Fast Flux

Zum Erkennen von Fast Flux wird Grubbs Test auf drei Eigenschaften angewendet. Dazu gehören die Anzahl von IPs auf die eine Domain über die Zeit auflöst, die Anzahl der Country-Codes einer Domain und die Anzahl der Besitzer einer Domain. Zuerst werden für jede Domain diese 3 Eigenschaften ermittelt. Anschließend wird Grubbs Test auf jeder Domain für allen 3 Eigenschaften durchgeführt. Wenn für eine Domain alle drei Eigenschaften Ausreißer sind, wird dies in der Logdatei vermerkt.

4.7.2 DGA

Zum Erkennen von Domain-Generation-Algorithms werden hier zwei Methoden benutzt. Eine einfache Methode ist das suchen nach NX Fehlern die in den passiven DNS-Daten gespeichert wurden. Hierzu wird Grubbs Test verwendet. Aus Elasticsearch werden alle Domains und die dazugehörigen Flags abgefragt und dann der Prozentsatz aller NX Fehler, für einzelne Tage und Wochen ermittelt. Diese zwei Datensätze können anschließend mit Hilfe von Grubbs Test auf Ausreißer untersucht werden. Der gesamte Prozess ist komplett automatisch. Nutzer/innen haben die Möglichkeit über eine manuelle Veränderung der Irrtumswahrscheinlichkeit Einfluss zu nehmen. Beim Fund eines Ausreißers wird dieser der aktuellen Logdatei hinzugefügt.
Die zweite Methode, die hier angewendet wird ist ein Random Forest. Hierbei werden von den passiven DNS-Daten nur die Domains verwendet. Zusätzlich wird eine Liste von Domains benötigt, von denen man ausgeht das sie keine DGAs sind und ein Wörterbuch der englischen Sprache. Beides muss in Form einer Textdatei vorliegen. Nachdem der Random Forrest trainiert ist, können dann die einzelen aufgezeichneten Domains ausgewertet werden. Dies erfolgt ebenfalls automatisch. Beim Auffinden einer verdächtigen Domain wird diese der aktuellen Logdatei hinzugefügt.

4.7.3 Phishing

Zur Detektion von Phishing wird zunächst die Levenshtein-Distanz verwendet. Dabei werden von den aufgezeichneten DNS-Daten ausschließlich die Top Level Domains(TLD) benötigt. Subdomains werden in diesem Fall nicht betrachtet. Der Grund hierfür ist das ein Besitzer einer Domain ohne Probleme Subdomins erstellen kann, die identisch zu Top Level Domains sind, die er nicht besitzt. Ein Angreifer hat also keine Motivation leicht veränderte Subdomains einzusetzten, was die Levenshtein-Distanz auf Subdomains nutzlos macht. Für die im überwachten Netzwerk aufgezeichneten TLDs wird je die Distanz zu allen zu überprüfenden TDLs ermittelt. Verschiedene Ansätze zum Aufbau einer solchen Liste von TDLs werden in Kapitel Mustererkennung unter Phishing gegeben. Um die Zahl der Fehlerkennungen niedrig zu halten bietet sich an, nur Domains zu melden bei denen die Distanz drei oder kleiner war. Außerdem kommt eine Whitelist zum Einsatz. So wird verhindert das einmal fälschlich als Phishing Domains erkannte Domains erneut irrtümlich erkannt werden. Die Auswertug findet hier also auch automatisch statt. Dazu kann die in Abbildung3.7 gezeigte Implementation genutzt werden. Nutzer/innen haben allerdings die Möglichkeit auf die Auswertung Einfluss zu nehmen, sowohl über die Schwelle ab welcher Distanz eine Domain gemeldet werden soll, als auch über eine Whitelist von Domains die selbst bei Erkennung nicht gemeldet werden.
Als zweite Methode zum Erkennen von Phishing wird das Suchen nach Substrings in Domains eingesetzt. Zunächst werden sowohl Punkte wie auch Kommata aus den Domains entfernt. Danach wird jede Domain nach einer Liste von Zeichenketten durchsucht. Zum Erstellen dieser Liste gibt es verschiedene Möglichkeiten, auf die im vohrigen Kapitel schon eingegangen wurde. Diese Methode wird komplett automatisch ausgeführt. Nutzer/innen haben allerdings die Möglichkeit Einfluss zu nehmen, über eine Whitelist von Domains die als unbedenklich gelten und über die Liste von Zeichenketten nach denen in den Domains gesucht werden soll.

4.7.4 Mail Spam

Zum Detektieren ob vom überwachten Netzwerk aus Mail Spam verschickt wird, bietet sich die Verwendung von Grubbs Test an. Dabei soll das Versenden von ungewöhnlich vielen Mails aus einem Netzwerk detektiert werden. Da Grubbs Test verwendet wird ist es wichtig das die Testdaten durch eine Normalverteilung abgebildet werden können. Es macht daher wenig Sinn die aufsummierten Mail-Exchange-Resource-Records für jeden Tag zu übergeben. Wenn diese Daten zum Beispiel. aus einem Firmennetzwerk stammen wird man am Wochenende eher weniger Aktivität haben als an einem Werktag. Daher macht es Sinn einzelne Wochentage zu vergleichen oder die MX Records einer ganzen Woche zu summieren und diese anschließend

auszuwerten. Die Auswertung findet in diesem Fall also automatisch statt. Nutzer/innen können jedoch durch das Verändern der Irrtumswahrscheinlichkeit Einfluss auf die Auswertung nehmen.

4.8 Logging

Für jeden Tag wird eine Logdatei erstellt. Bei einem Fund der Auswertungskomponente wird dieser in einer Logdatei gespeichert. Dabei werden alle passiven DNS-Daten gespeichert die zum jeweiligen Datenpaket gehören.

4.9 Report

Das Melden eines Fundes findet per E-Mail statt. Dazu wird Mutt genutzt, ein textbasierter E-Mail-Client. Mit diesem wird dem Nutzer alle 24 Stunden die Logdatei des Tages geschickt, sofern es über den Tag Funde gab. Mutt ermöglicht außerdem das Verschlüsseln der E-Mail mit Pretty Good Privacy. Das verschicken wird mit Hilfe eines Cronjobs automatisiert.

5 Zusammenfassung und Ausblick

5.1 Was wurde erarbeitet?

Diese Arbeit hat zunächst einen Überblick über DNS und im Besonderen Passiv-DNS gegeben. Der Schwerpunkt lag hierbei auf Informationen, die für das Finden von Angriffen in Passiv-DNS-Daten relevant sind. Dabei wurden Angriffsmuster, die sich in Passiv-DNS Daten finden erläutert.
Anschließend wurden alle nötigen Elemente von Datensammlung, über die Datenverarbeitung, bis zur Mustererkennung für das Finden von Angriffen in Passiv-DNS-Daten vorgestellt. Dabei wurde ein Verständnis für die Nutzung von Passiv-DNS im Umfeld der IT-Sicherheit geschaffen.

5.2 Wurde das Ziel erreicht?

Das Ziel dieser Arbeit war es, das Potential von DNS und im Besonderen Passiv-DNS im Bereich der IT-Sicherheit aufzuzeigen. Dabei wurden Angriffe, die sich in DNS-Daten erkennen lassen vorgestellt. Somit wurde das in 1.2 definierte Ziel der Arbeit erreicht.

5.3 Wo geht es weiter?

Passiv-DNS bietet ein hohes Potential im Themenbereich der IT-Sicherheit. Dieses wurde natürlich in dieser Arbeit nicht vollständig ausgeschöpft. Allgemein könnte man alle vorgestellten Testmethoden noch weiter im Detail bearbeiten. Dabei sollte die Minimierung der Fehlerrate ein Hauptaugenmerk sein, da einserseits dies das qualitative Hauptmanko darstellt und andererseits hierdurch die größte personelle Entlastung erzielt werden könnte. Um repräsentativere und validere Daten erhalten zu können, würde sich die Verwendung von Daten aus passiven DNS-Datenbanken für weiteres Testen der vorgestellten Methoden anbieten. In herausragender Weise sollte angemerkt sein, dass bei Fast Flux das Problem einer Verwechslung mit Content Delivery Networks besteht. Im Kontext von DGAs sollte der Random Decision Forest um zusätzliche Sprachen erweitert werden, weil nur so die lexikographischen Unterschiede berücksichtigt werden können. Weiterhin haben verschiedene DGAs unterschiedliche Algorithmen

zur Generierung von Domainnamen, weswegen auch Tests gegen Domains weiterer DGAs sinnvoll wären. Was das Phishing betrifft könnte man die hier vorgestellten Methoden in Form einer Browsererweiterung implementieren. Dies hätte den Vorteil, dass Nutzer/innen gewarnt würden bevor Schaden entstehen kann. Im Rahmen der Ausreißeranalyse wäre es zusätzlich sinnvoll Normalwerte für relevante DNS Werte zu bestimmen. Schlussendlich ist IT-Sicherheit ein Themenfeld was - nicht nur im Fachkontext der Informationswissenschaften - sondern gerade im Alltag des Internets der Dinge, wegen seiner beispiellosen gesellschaftlichen Relevanz, auch weiterhin eine hohe Aufmerksamkeit verdient.

Literaturverzeichnis

[1] Alain Aina, Jaap Akkerhuis, KC Claffy, Steve Crocker, Daniel Karrenberg, Johan Ihren, Rodney Joffe, Mark Kosters, Allison Mankin, Ram Mohan, Russ Mundy, Frederico Neves, Jon Peterson, David Piscitello, Ray Plzak, Mike St. Johns, Doron Shikmoni, Bruce Tonkin, Paul Vixie, and Suzanne Woolf. Ssac advisory sac008. https://www.icann.org/en/system/files/files/dns-ddos-advisory-31mar06-en.pdf. Abgerufen am 28.03.2017.

[2] M. Andrews. Negative caching of dns queries (dns ncache)). https://tools.ietf.org/html/rfc2308. Abgerufen am 28.03.2017.

[3] Lars Are Aschim. Free and secure public dns service. https://www.mnemonic.no/news/2015/free-and-secure-public-dns-service/. Abgerufen am 28.03.2017.

[4] D. Atkins. Threat analysis of the domain name system. https://tools.ietf.org/html/rfc3833. Abgerufen am 21.03.2017.

[5] D. Atkins and R. Austein. Threat analysis of the domain name system. https://tools.ietf.org/html/rfc3833. Abgerufen am 28.03.2017.

[6] D. Barr. Common dns operational and configuration errors. https://tools.ietf.org/html/rfc1912. Abgerufen am 22.03.2017.

[7] BFK. Bfk passive dns replication. http://www.bfk.de/bfk_dnslogger.html. Abgerufen am 28.03.2017.

[8] Leyla Bilge. Exposure: Finding malicious domains using passive dns analysis. https://www.isoc.org/isoc/conferences/ndss/11/slides/4_2.pdf. Abgerufen am 20.03.2017.

[9] Benjamin Braun. Investigating dns hijacking through high frequency measurements. https://escholarship.org/uc/item/8tm5c7r7#page-4. Abgerufen am 28.03.2017.

[10] BSI. Dns-grundbegriffe. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02450.html. Abgerufen am 21.03.2017.

[11] BSI. Einsatz von dnssec. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04353.html. Abgerufen am 21.03.2017.

[12] BSI. Ip-adresse ipv4, ipv6. https://www.bsi.bund.de/SharedDocs/Glossareintraege/DE/I/IP-Adresse.html. Abgerufen am 21.03.2017.

[13] BustSpammers.com. Get smart on phishing! learn to read links! http://www.bustspammers.com/phishing_links.html. Abgerufen am 23.03.2017.

[14] Elasticsearch BV. Elasticsearch. https://www.elastic.co/. Abgerufen am 28.03.2017.

[15] CIRCL. Circl passive dns. https://www.circl.lu/services/passive-dns/. Abgerufen am 28.03.2017.

[16] DAMBALLA. Dgas in the hands of cyber-criminals. https://www.damballa.com/downloads/r_pubs/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf. Abgerufen am 22.03.2017.

[17] Dglosser. Bh dns files. http://www.malwaredomains.com/?page_id=66. Abgerufen am 23.03.2017.

[18] Ivan Dimov. Phishing techniques: Similarities, differences and trends. http://resources.infosecinstitute.com/phishing-techniques-similarities-differences-and-trends-part-i-mass-phishing/. Abgerufen am 23.03.2017.

[19] Rudolf Felser. Angriffe auf it-sicherheit steigen, budgets sinken. http://www.computerwelt.at/news/technologie-strategie/security/detail/artikel/107010-angriffe-auf-it-sicherheit-steigen-budgets-sinken/. Abgerufen am 20.03.2017.

[20] Edward Bjarte Fjellskal. passivedns. https://github.com/gamelinux/passiveDNS. Abgerufen am 22.03.2017.

[21] Wireshark Foundation. tshark. https://www.wireshark.org/docs/man-pages/tshark.html. Abgerufen am 24.03.2017.

[22] Wireshark Foundation. Wireshark. https://www.wireshark.org/. Abgerufen am 24.03.2017.

[23] R. Gieben and W. Mekking. Authenticated denial of existence in the dns. https://www.heise.de/netze/rfc/rfcs/rfc7129.shtml. Abgerufen am 04.04.2017.

[24] Dan Goodin. Fake subpoenas harpoon 2,100 corporate fat cats. https://www.theregister.co.uk/2008/04/16/whaling_expedition_continues/. Abgerufen am 23.03.2017.

[25] Frank E. Grubbs. The annals of mathematical statistics. https://projecteuclid.org/euclid.aoms/1177729885. Abgerufen am 25.03.2017.

[26] Frank E. Grubbs. Procedures for detecting outlying observations in samples. http://www.tandfonline.com/doi/abs/10.1080/00401706.1969.10490657. Abgerufen am 24.03.2017.

[27] Tin Kam Ho. Random decision forests. http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf. Abgerufen am 25.03.2017.

[28] Linh Vu Hong. Dns traffic analysis for network-based malware detection. http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/6309/pdf/imm6309.pdf. Abgerufen am 20.03.2017.

[29] Linh Vu Hong. Dns traffic analysis for network-based malware detection. http://nordsecmob.aalto.fi/en/publications/theses_2012/vu_hong-linh_thesis.pdf. Abgerufen am 23.03.2017.

[30] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for dns over transport layer security (tls). https://tools.ietf.org/html/rfc7858/. Abgerufen am 31.03.2017.

[31] Endgame Inc. Dga classifier. https://github.com/endgameinc/SANS_THIR16. Abgerufen am 25.03.2017.

[32] Endgame Inc. Hunting on networks, part 2: Higher-order patterns. https://www.endgame.com/blog/hunting-networks-part-2-higher-order-patterns. Abgerufen am 15.03.2017.

[33] Farsightsecurity Inc. Farsightsecurity order services. https://www.farsightsecurity.com/order-services/. Abgerufen am 28.03.2017.

[34] Internet Solutions Inc. Hosts file. http://support.isoc.net/Page.aspx/117/hosts.html. Abgerufen am 21.03.2017.

[35] OpenDNS Inc. Opendns 2010 report. https://www.scribd.com/document/48081517/Opendns-Report-2010. Abgerufen am 9.04.2017.

[36] Kaspersky. What is kaspersky anti-virus ndis filter in kaspersky anti-virus version 6.0.4.x. https://support.kaspersky.com/8596. Abgerufen am 26.04.2017.

[37] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf. Abgerufen am 23.03.2017.

[38] Cricket Liu. Strengthen your network security with passive dns. http://www.infoworld.com/article/2994016/network-security/strengthen-your-network-security-with-passive-dns.html. Abgerufen am 26.04.2017.

[39] Jerry Lundstrom. Dnscap. https://github.com/verisign/DNScap. Abgerufen am 24.03.2017.

[40] P. C. Mahalanobis. On the generalized distance in statistics. http://insa.nic.in/writereaddata/UpLoadedFiles/PINSA/Vol02_1936_1_Art05.pdf. Abgerufen am 10.04.2017.

[41] Dhia Mahjoub. Finding the patterns in a mysterious new dga. https://umbrella.cisco.com/blog/blog/2013/10/24/mysterious-dga-lets-investigate-sgraph/. Abgerufen am 24.03.2017.

[42] Cade Metz. Why does the net still work on christmas? paul mockapetris. http://internethalloffame.org/blog/2012/07/23/why-does-net-still-work-christmas-paul-mockapetris. Abgerufen am 21.03.2017.

[43] Microsoft. Resource record types. https://technet.microsoft.com/de-de/library/cc958958.aspx. Abgerufen am 21.03.2017.

[44] Paul Mockapetris. Rfc 1034 - domain names - concepts and facilities. https://www.ietf.org/rfc/rfc1034.txt. Abgerufen am 21.03.2017.

[45] Paul Mockapetris. Rfc 1035 - domain names - implementation and specification. https://www.ietf.org/rfc/rfc1035.txt. Abgerufen am 21.03.2017.

[46] Paul Mockapetris. Rfc 882 - domain names - concepts and facilities. https://tools.ietf.org/html/rfc882. Abgerufen am 21.03.2017.

[47] Paul Mockapetris. Rfc 883 - domain names - implementation and specification. https://tools.ietf.org/html/rfc883. Abgerufen am 21.03.2017.

[48] Mozilla. Idn display algorithm. https://wiki.mozilla.org/IDN_Display_Algorithm#Other_Browsers. Abgerufen am 24.03.2017.

[49] Paul Mutton. Fraudsters seek to make phishing sites undetectable by content filters. https://news.netcraft.com/archives/2005/05/12/fraudsters_seek_to_make_phishing_sites_undetectable_by_content_filters.html. Abgerufen am 24.03.2017.

[50] NeutralizeThreat. Nivdort code obfuscation and dga. http://www.neutralizethreat.com/2015/12/nivdort-code-obfuscation-and-dga.html. Abgerufen am 06.04.2017.

[51] ITL NIST. Grubbs test for outliers. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35h1.htm. Abgerufen am 25.03.2017.

[52] Tom Olzak. Dns cache poisoning: Definition and prevention. http://adventuresinsecurity.com/Papers/DNS_Cache_Poisoning.pdf. Abgerufen am 28.03.2017.

[53] Dave Piscitello. 5 ways to monitor dns traffic for security threats. http://www.darkreading.com/analytics/threat-intelligence/5-ways-to-monitor-dns-traffic-for-security-threats/a/d-id/1315868. Abgerufen am 26.04.2017.

[54] Matthew Prince. The ddos that knocked spamhaus offline (and how we mitigated it). https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/. Abgerufen am 28.03.2017.

[55] The Chromium Project. Idn in google chrome. https://www.chromium.org/developers/design-documents/idn-in-google-chrome. Abgerufen am 24.03.2017.

[56] Jamie Riden. How fast-flux service networks work. https://www.honeynet.org/node/132. Abgerufen am 22.03.2017.

[57] RiskIQ. riskiq.com passive dns. https://www.riskiq.com/products/security-intelligence-services/passive-dns/. Abgerufen am 28.03.2017.

[58] Farsight Security. Farsight dns sensor. https://github.com/farsightsec/sie-DNS-sensor. Abgerufen am 24.03.2017.

[59] Farsight Security. farsightsecurity.com. https://www.farsightsecurity.com/. Abgerufen am 22.03.2017.

[60] Farsight Security. Passive dns faq. https://www.farsightsecurity.com/technical/passive-dns/passive-dns-faq/. Abgerufen am 22.03.2017.

[61] Ryan Singel. Isps error page ads let hackers hijack entire web, researcher discloses). https://www.wired.com/2008/04/isps-error-page. Abgerufen am 28.03.2017.

[62] Peter Stavroulakis, Al Fayyadh Bander, Grandison Tyrone, Alzomai Mohammed, and McNamara Juditha. Security usability principles for vulnerability analysis and risk assessment. http://eprints.qut.edu.au/18542/. Abgerufen am 23.03.2017.

[63] Peter Stavroulakis and Mark Stamp. Handbook of information and communication security. https://books.google.nl/books?id=I-9P1EkTkigC&pg=PA433&redir_esc=y. Abgerufen am 23.03.2017.

[64] Debbie Stephenson. Spear phishing: Who's getting caught? https://www.firmex.com/thedealroom/spear-phishing-whos-getting-caught//. Abgerufen am 23.03.2017.

[65] Brian Trammell, Chris Inacio, Michael Duggan, Emily Sarneso, Dan Ruef, and the CERT Network Situational Awareness Group Engineering Team. Yaf. http://Tools.netsa.cert.org/yaf/yaf.html. Abgerufen am 24.03.2017.

[66] US-CERT. Dns amplification attacks. https://www.us-cert.gov/ncas/alerts/TA13-088A. Abgerufen am 28.03.2017.

[67] Virustotal. Searching with virustotal. https://www.virustotal.com/en/documentation/searching/. Abgerufen am 28.03.2017.

[68] Webdevmedia. tcpiputils.com premium api access. http://www.tcpiputils.com/api. Abgerufen am 28.03.2017.

[69] Florian Weimer. Florian weimar: Passive dns replication. http://www.enyo.de/fw/software/dnslogger/first2005-paper.pdf. Abgerufen am 22.03.2017.

[70] Florian Weimer. Passive dns replication. http://www.enyo.de/fw/software/dnslogger/first2005-paper.pdf. Abgerufen am 26.04.2017.

[71] Wikipedia. Dns-raum. https://de.wikipedia.org/wiki/Datei:Dns-raum.svg. Abgerufen am 22.03.2017.

[72] Wikipedia. Fast-flux.png. https://de.wikipedia.org/wiki/Datei:Fast-Flux.png. Abgerufen am 23.03.2017.

[73] Wei Xu, Kyle Sanders, and Yanxin Zhang. We know it before you do: Predicting malicious domains. https://www.virusbulletin.com/uploads/pdf/conference/vb2014/VB2014-XuZhangSanders.pdf. Abgerufen am 23.03.2017.

[74] Sousan Yazdi. A closer look at cryptolocker's dga. https://blog.fortinet.com/2014/01/16/a-closer-look-at-cryptolocker-s-dga. Abgerufen am 23.03.2017.

[75] Igal Zeifman. Dns flood of 1.5 billion requests a minute, fueled by ddos protection services. https://www.incapsula.com/blog/massive-dns-ddos-flood.html. Abgerufen am 28.03.2017.

[76] Jiong Zhang and Mohammad Zulkernine. Anomaly based network intrusion detection with unsupervised outlier detection. https://www.ccsl.carleton.ca/mmcna.old/publications/NS0907.pdf. Abgerufen am 24.03.2017.

Glossar

Anycast Anycast ist eine Netzwerkadressierungs und Routing Methode bei der Datagramms von einem einzelnen Sender zum topologisch nächsten Empfänger gesendet werden.

Botnet Ein Botnet ist eine Anzahl von Geräten die mit dem Internet verbunden sind und vom Besitzer des Botnet kontrolliert werden. Sie werden unter andrem für Distributed-Denial-Of-Service-Attacks, stehlen von Daten und zum Versenden von Spam genutzt.

Cross-Site-Scripting Cross-Site-Scripting (XXS) ist eine Computersicherheitslücke. XXS ermöglicht Angreifern das Injizieren von Client seitigen Skripten in Webseiten.

DGA Ein Domain-Generation-Algorithm oder kurz DGA ist ein Algorithmus, der von Maleware genutzt wird um periodisch große Mengen von Domainnamen zu generieren.

Fast Flux Fast-Flux ist eine Technik die von Botnetzen zum Verschleiern von Maleware verteilenden Webseiten genutzt wird. Seiten werden hinter einem sich ständig ändernden Netzwerk aus Hosts versteckt, die als Proxy dienen.

IETF Die Internet-Engineering-Task-Force kurz IETF ist eine Organisation, die die freiwillige Nutzung von Internet Standards fördert.

ISP Ein Internet-Service-Provider, kurz ISP ist eine Organisation die Dienste zur Nutzung des Internets anbietet.

IT Informationstechnik, kurz IT ist die Anwendung von Computern zum speichern, übermitteln und manipulieren von Daten.

Loopback-Interface Ein Loopback-Interface ist eine virtuelle Netzwerkschnittstelle durch die Netzwerkanwendungen kommunizieren können, sofern sie sich auf der gleichen Maschine befinden.

PTR Ein Pointer-Record kurz PTR ist ein Zeiger auf einen Canonical Name Record kurz CNAME.

Reverse Lookup Ein Reverse-DNS-Lookup ist eine Anfrage an DNS die zu einem Domainnamen die IP-Adresse zurück liefert.

RFC Request-For-Comments kurz RFC ist eine Publikation der IETF. RFC werden zur Peer-Review oder zur Vermittlung von neuen Konzepten oder Informationen genutzt. Einige der RFC werden von der IETF als Internet Standards adaptiert.

Social Engineering Social-Engineering im Kontext von Informationssicherheit bezieht sich auf die psychologische Manipulation von Menschen, um sie dazu zu bewegen bestimmte Handlungen auszuführen oder Informationen preiszugeben.

TTL Time-To-Live kurz TTL ist ein Mechanismus um die Lebensspanne von Daten in einem Computernetzwerk zu beschränken.

Whois Whois ist ein Protokoll das eingesetzt wird um Anfragen an Datenbanken zu stellen, die Informationen über Internetressourcen wie z.B. wem eine Domain gehört speichern.

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

Hamburg, 28. April 2017 Nick Diedrich