Technische Universität München

Fakultät für Informatik

Bachelor's Thesis in Informatik

Botnet-Evaluierung und Botnet-Erkennung

Sebastian Kiwus

Technische Universität München

Fakultät für Informatik

Bachelor's Thesis in Informatik

Botnet-Evaluierung und Botnet-Erkennung

Botnet Evaluation and Botnet Detection

Autor	Sebastian Kiwus
Aufgabensteller	Prof. Dr.-Ing. Georg Carle
Betreuer	Nadine Herold, M.Sc; Dipl.-Inf. Stephan-A. Posselt
Datum	29. September 2015

Informatik VIII
Lehrstuhl für Netzarchitekturen und Netzdienste

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

Garching b. München, 29. September 2015

Unterschrift

Abstract

The use of botnets for financial or political attacks is gaining in popularity. As a consequence, the importance of detection programs increases. The resulting arms race between botnets and detection programms leads to an increasing complexity of both. To determine, whether various detection programs differ in their suitability for detecting different botnets, a definition of characteristics and criteria is required, on which the programs can be classified and compared.

This work defines characteristics in which botnets and detection programs can differ. Furthermore, criteria are specified, which can be used to compare different aspects of botnets and detection programs. For their determination, various metrics are provided.

Botnets are integrated in a scientific test environment in order to enable a evaluation of different detection programs with equal test conditions.

To evaluate the defined criteria, specific botnets and detection programs are selected, which are presented and analyzed in detail. The selected botnets are *BeEF* and *Zeus*. For evaluation of the detection programs *Snort*, *Bro*, and *Antidoto* are used.

Based on the previously defined criteria the evaluation results of botnets and detection programs are interpreted in terms of their impact on detection.

Zusammenfassung

Die Benutzung von Botnetzen für finanzielle oder politische Angriffe gewinnt zunehmend an Popularität. Dadurch steigt auch die Bedeutung von Erkennungsprogrammen zur Abwehr dieser. Das daraus resultierende Wettrüsten zwischen Botnetzen und Programmen zu deren Abwehr führt zu einer steigenden Komplexität beider Seiten. Um zu überprüfen, ob sich Erkennungsprogramme in ihrer Eignung zur Erkennung verschiedener Botnetze unterscheiden, ist eine Definition von Merkmalen und Kriterien erforderlich, nach denen die Programme eingeordnet und untereinander verglichen werden können.

Diese Arbeit definiert Merkmale, in denen sich Botnetze und Erkennungsprogramme unterscheiden können. Außerdem werden Kriterien festgesetzt, welche für einen Vergleich verschiedener Aspekte der Botnetze und Erkennungsprogramme verwendet werden können und stellt Möglichkeiten zu deren Bestimmung vor.

In einer wissenschaftlichen Testumgebung werden Botnetze integriert, die eine Evaluierung verschiedener Erkennungsprogramme unter Gewährleistung gleicher Testbedingungen ermöglicht.

Zur Evaluierung der definierten Kriterien werden gezielt Botnetze und Erkennungsprogramme ausgewählt, welche einzeln vorgestellt und analysiert werden. Bei den Botnetzen kommen dabei *BeEF* und *Zeus* zum Einsatz, bei den Erkennungsprogrammen *Snort*, *Bro*, und *Antidoto*.

Anhand der zuvor definierten Kriterien werden die Evaluierungsergebnisse der Botnetze und Erkennungsprogramme bezüglich ihrer Auswirkungen auf eine Erkennung interpretiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Mit der voranschreitenden Technisierung gewinnt auch das Thema Schadsoftware zunehmend an Bedeutung. Eine Gruppe von Schadsoftware bilden die Botnetze, deren Ziel die Infizierung fremder Rechner mit ihrem Schadcode darstellt. Anschließend warten sie im Verborgenen auf einen Befehl, der sie aktiv werden lässt.

Die Infektion eines Rechners kann auf unterschiedlichen Wegen stattfinden, zum Beispiel durch das Anklicken eines Links auf einer Website oder durch das Öffnen des Anhangs einer E-Mail. Sobald eine entsprechende Aktion ausgelöst wird, lädt sich die Botnetz-Software auf den Rechner herunter und infiziert ihn. Der rekrutierte Rechner wird daraufhin als *Zombie* oder *Bot* bezeichnet.

Die Gesamtheit aller zusammengehörigen Bots bilden ein Botnetz. Ein solches Botnetz kann mehrere Millionen infizierte Rechner enthalten. Das Botnetz Bredolab beispielweise wurde im Jahr 2010 entdeckt und bestand aus ungefähr 30 Millionen Bots. [1] Die Größe eines Botnetzes ist von entscheidender Bedeutung, da das Schadenspotenzial mit dieser skaliert.

Die Architektur der meisten Botnetze ist hierarchisch. Dabei existiert ein Command-and-Control-Server, welcher vom Angreifer direkt gesteuert wird und zur Kontrolle des gesamten Botnetzes dient. Soll ein Angriff auf ein beliebiges Ziel gestartet werden, übermittelt der Server die entsprechenden Befehle an die Bots, welche dann die eigentliche Aktion ausführen. Die Abschaltung eines solchen Botnetzes ist meist nur durch Deaktivierung des C&C-Servers möglich. Die Eliminierung einzelner Bots hat nur einen sehr geringen Einfluss, da das Botnetz trotz Ausfall einzelner Bots weiter agieren kann. Dies stellt ein wesentliches Problem für Abwehrmechanismen dar, weil der Kontrollserver nur indirekt an einem Angriff teilnimmt und somit eine Rückverfolgung oft nur zu einzelnen Bots, jedoch nicht zum Server führt. Der Angreifer kann seine Aktivitäten noch weiter anonymisieren, indem der C&C-Server nicht direkt mit den Bots kommuniziert, sondern über Bot-Controller. Dies sind Server, welche in der Hierarchie zwischen dem C&C-Server und den Bots stehen und die Befehle vom C&C-Server an die Bots

weiterleiten. Eine beispielhafte Illustration eines Botnetz-Angriffs mit Bot-Controllern ist in Abbildung 1.1 dargestellt.

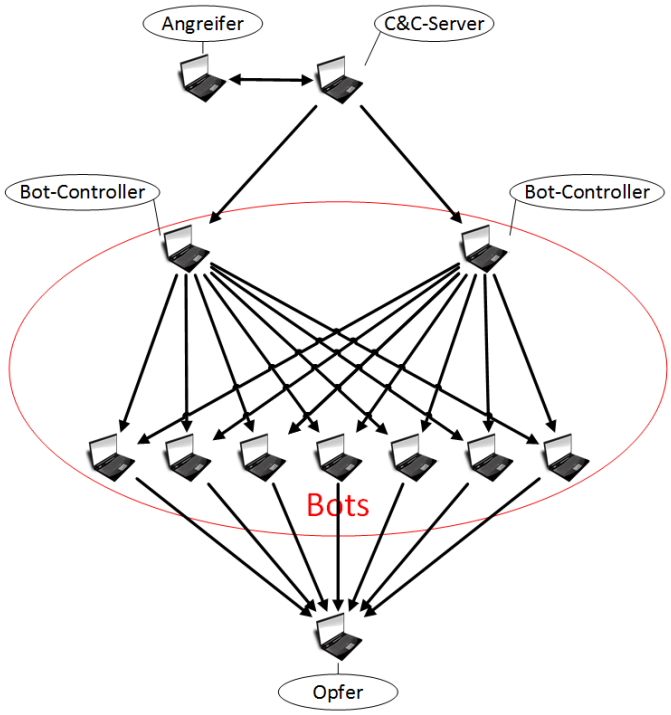

Abbildung 1.1: Angriffsbeispiel eines Botnetzes mit Bot-Controllern

Da Botnetze versuchen, ihre Aktivitäten möglichst unauffällig auszuüben, ist die Erkennung einer Botnetz-Infizierung nicht einfach. Es existieren zahlreiche Erkennungsprogramme, welche auf verschiedene Arten versuchen, Botnetze zu erkennen.

Neben der hohen Priorisierung der Verborgenheit kann zwischen drei weiteren Zielen der Botnetze differenziert werden: Verbreitung, Sammeln von Informationen und Verarbeitung von Informationen. Diese Ziele sind eng miteinander verbunden. Wurde ein Rechner mit der Schadsoftware infiziert, kann er seine Umgebung nach Schwachstellen scannen, welche für die Infizierung weiterer Rechner verwendet werden können. Sind keine Schwachstellen in der Umgebung vorhanden, werden die Ergebnisse der Scans für einen eventuellen späteren Missbrauch gesammelt und an den Betreiber des Botnetzes weitergeleitet. [2]

Botnetze können für verschiedene Zwecke wie das Verschicken von Spam-Mails, Installieren von Schadsoftware oder wie bereits erwähnt zur Ausspähung benutzt werden. Hauptverwendungszweck stellen jedoch Distributed-Denial-of-Service Angriffe dar. Dies sind Angriffe, welche versuchen, ein System zu überlasten, indem eine große Anzahl legitim aussehender Anfragen an das Ziel geschickt wird, bis es aufgrund des hohen Datenaufkommens nicht mehr handlungsfähig ist. Dadurch ist die Bearbeitung

legitimer Anfragen nicht mehr möglich und führt zu Verwerfungen. Handelt es sich beim Opfer beispielweise um einen Webserver, ist als Ergebnis des Angriffs die Webseite nicht mehr erreichbar. Ein Botnetz eignet sich für derartige Angriffe aus mehreren Gründen: Da es meist aus einer Vielzahl an Bots besteht, werden Anfragen nicht von einem einzelnen Rechner, sondern vom gesamten Botnetz verschickt. Dies sorgt für ein größeres Datenaufkommen und eine stärkere Überlastung des Opfers. Desweiteren haben die Versender der Pakete nicht nur eine einzige, sondern unterschiedliche IP-Adressen. Da die Verbannung einer Vielzahl von IP-Adressen ebenfalls zu einem hohen Ressourcenaufwand führt, ist eine Abwehr zusätzlich erschwert. Den Hauptgrund für eine Verwendung stellt aber die für den Angreifer gewährleistet Anonymität dar, weil er nur indirekt an einem Angriff teilnimmt. [3]

Aufgrund einer erschwerten Abwehr von Angriffen mittels Botnetzen, führte dies zu einer stark wachsenden Popularität bei Kriminellen. Messungen aus dem Jahr 2009 zeigen, dass bis zu 90% des gesamten E-Mail Verkehrs Spam-Mails sind. [4]

Erst Anfang dieses Jahres wurden die Internetseiten des Bundestags und Angela Merkels Opfer eines DDoS-Angriffes, wodurch sie zeitweise nur eingeschränkt zugänglich waren. [5] Dies zeigt, dass Botnetze inzwischen nicht mehr nur für finanzielle, sondern auch für politische Ziele genutzt werden. Da derartige Angriffe auch für terroristische Absichten verwendet werden können, stellen sie für Regierungen eine zunehmende Gefahr dar und führen zu einer militärischen Erforschung der Thematik um Botnetze. So war das US-Army Research Office (ARO) unter anderem Hauptsponsor des Botnetz-Erkennungsprogrammes Bothunter. [2]

1.1 Zielsetzung der Arbeit

Im Rahmen dieser Arbeit werden sowohl Programme zur Bildung eines Botnetzes, als auch Programme zur Erkennung oder Abwehr dieser untersucht. Dafür werden verschiedene Klassifizierungsmerkmale für Botnetze und Erkennungsprogramme definiert, mit denen die Programme eingeordnet werden können, da für die Erkennung unterschiedlicher Botnetze auch verschiedene Ansätze verwendet werden können.

Außerdem sollen verschiedene Kriterien festgesetzt werden, welche für einen Vergleich verschiedener Aspekte der Botnetze und Erkennungsprogramme verwendet werden können. Zur Bestimmung dieser Kriterien sollen spezifische Experimente definiert werden.

Desweiteren sollen in einer wissenschaftlichen Testumgebung Botnetze und Erkennungsprogramme integriert werden. Anschließend erfolgt mit den definierten Kriterien eine Evaluierung der verwendeten Programme und eine Interpretierung der Ergebnisse

1.2 Gliederung der Arbeit

Zu Beginn der Arbeit werden die festgesetzten Ziele vorgestellt, welche mit dieser Arbeit erreicht werden sollen.

In Kapitel 2 werden verwandte Arbeiten vorgestellt. Diese werden bezüglich ihrer Thematik nach Botnetz-Software und Erkennungsprogrammen unterteilt.

In Kapitel 3 erfolgt die Analyse von Botnetz-Software und Erkennungsprogrammen. Dabei werden für beide Kategorisierungsmerkmale vorgestellt, mit denen sich die Programme bezüglich ihren Merkmalen einordnen lassen. Außerdem werden die zur Evaluierung ausgewählten Programme analysiert und beschrieben. Zuletzt werden die Evaluationskriterien für Botnetze und Erkennungsprogramme vorgestellt.

Nach der Analyse erfolgt in Kapitel 4 die Beschreibung der Testumgebung. Dabei wird das benutzte Testbed präsentiert und neben dem Aufbau der verwendeten Rechner die Experimente zur Bestimmung der Evaluationskriterien vorgestellt.

In Kapitel 5 werden anhand der definierten Kriterien die ermittelten Daten analysiert und interpretiert. Im letzten Abschnitt dieser Arbeit werden die Ergebnisse zusammengefasst.

Kapitel 2

Related Work

Im Folgenden werden einige verwandte Arbeiten vorgestellt, welche eine besondere Relevanz aufweisen. Dabei wird zwischen Arbeiten über Botnetz-Software und Botnetzerkennung unterschieden.

2.1 Botnetz Software

David Dagon et al. beschäftigt sich in [6] mit der Aufstellung einer Taxonomie für Botnetze. Dabei werden verschiedene Klassifizierungskriterien bezüglich der Nützlichkeit für den Botmaster bei unterschiedlichen Verwendungen beurteilt. Es wird festgestellt, dass die Topologie eines Botnetzes bei sehr großen Netzen keine entscheidende Rolle zur Bewertung der Nützlichkeit darstellt. Zwar kann die Topologie Einfluss auf Kriterien wie die Anonymität des Botmasters haben, dies ist aber von geringerer Bedeutung als das Design der Anwendungsschicht oder das gewählte Protokoll für den Nachrichtenaustausch. Die drei Kriterien der Effektivität, Effizienz und Robustheit werden als wichtigste Unterscheidungskriterien festgesetzt. Die wichtigsten Metriken zur Feststellung der Effektivität eines Botnetzes sind dessen Größe und die verfügbare Bandbreite. Dafür wird für jeden Bot die geschätzte Wahrscheinlichkeit berechnet, dass eine bestimmte Bandbreite gegeben ist. Zur Bestimmung der Effizienz wird die Kommunikation in einem Botnetz betrachtet. Diese kann durch Beurteilung der Netzwerklänge gemessen werden. Können Bots kurze Pfade zur Kommunikation nutzen, ist die Wahrscheinlichkeit einer Entdeckung geringer und eine höhere Effizienz gewährleistet. Um die Robustheit eines Botnetzes zu beurteilen wird ebenfalls die Länge der Kommunikationspfade betrachtet. Zusätzlich wird versucht, die Transitivität eines Botnetzes zu beurteilen. Bei einer höheren Transitivität führt das Ausfallen einzelner Bots zu einer geringeren Beeinträchtigung des Botnetzes. Durch die Vorstellung und Begründung diverser Unterscheidungskriterien und Möglichkeiten zu deren Messung werden Ansätze zur Evaluierung von Botnetzen bereitgestellt, welche in der eigenen Arbeit aufgegriffen

werden können.

In [7] beschäftigen sich Brigitte Lundeen et al. mit der Thematik des Clickjackings und präsentieren eine ausführliche Erläuterung zum verwendeten Botnetz-Programm BeEF. Ziel dieser Arbeit ist das Schreiben eines eigenen Moduls für BeEF, welches das Programm um die Funktion des Clickjackings erweitert. Dadurch wird ein Einblick in die API von BeEF ermöglicht und die Funktionsweise der von BeEF verwendeten Javaskripte zur Steuerung anderer Rechner vorgestellt. Bei Etablierung des Clickjacking-Moduls kann der Nutzer ohne sein Wissen zum Herunterladen eines Javascripts gebracht werden. Dieses kann in Verbindung mit BeEF zahlreichende Aktionen beinhalten. Durch die dokumentierte Arbeit mit dem Programm BeEF wird die eigene Analyse dieses Programms unterstützt.

Bei der Arbeit [8] hat das Team von Hamad Binsalleeh et al. ein Reverse Engineering des Zeus Botnetzes durchgeführt, um eine umfangreiche Analyse dieses Programms zu ermöglichen. Dies stellt eine große Relevanz dar, da das Zeus Botnetz auch in der eigenen Arbeit analysiert und evaluiert wird und somit vorgestellte Ansätze aufgegriffen werden können. Es werden Informationen über Zeus vermittelt, welche den Erfolg und die Gründe für dessen hohe Beliebtheit bei Hackern beschreiben. Entscheidend ist die ausführliche Analyse der in Zeus enthaltenen Komponenten. Dabei wird auch der Inhalt und die Funktion der benötigten Dateien wie diverse Konfigurationsdateien vorgestellt. Zusätzlich wird ein Schwerpunkt auf den Netzwerkverkehr von Zeus gelegt. Indem der Ablauf einer typischen Kommunikation zwischen dem Zeus C&C-Server und einem Bot herausgearbeitet wird, können Ansätze erkannt werden, mit denen Erkennungsprogramme die Kommunikationsmuster wiedererkennen können. Außerdem wird der Verlauf einer Botnetz-Infizierung mit der Anlegung verschiedener Einträge in den Systemdateien von Windows beschrieben. Diese Informationen können für die Analyse von Erkennungsprogrammen verwendet werden, indem Signatur-basierte Programme Zeus durch Überprüfung dieser Systemdateien erkennen.

2.2 Botnetzerkennung

In [9] wird ein mögliches Klassifizierungsschema für Botnetz-Erkennungsprogramme präsentiert, indem die Verfahren aller bekannten Botnetzerkennungsmechanismen nach bestimmten Kriterien unterschieden werden. Zunächst erfolgt dabei die Unterteilung in Intrusion Detection Systeme (IDS) und Honeypots. Da in der eigenen Arbeit ausschließlich Programme der ersten Kategorie verwendet werden, sind nur diese für eine Analyse und Klassifizierung von Interesse. Intrusion Detection Systeme werden in Signatur-basierte und Anomalie-basierte Ansätze unterteilt. Bei zweiteren wird zusätzlich zwischen Host-basierten und Netzwerk-basierten Techniken differenziert. Außerdem kann bei Netzwerk-basierten Anomalie-Erkennungen ein Netzwerk entweder

aktiv oder passiv überwacht werden. Zu jedem Unterpunkt wird eine Beschreibung gegeben und die Stärken beziehungsweise Grenzen einer Verwendung in einem Erkennungsprogramm verdeutlicht. Außerdem wird das Kriterium der Falsch-Positiv-Rate zur Evaluation von Erkennungsprogrammen vorgestellt.

Ein ähnlicher Ansatz zur Klassifizierung von Erkennungsprogrammen wird in [1] aufgegriffen. Hierbei werden Anomalie-basierte Verfahren nicht weiter unterteilt. Zusätzlich werden kombinierte Verfahren vorgestellt, welche nicht nur ein einziges Verfahren, sondern mehrere gleichzeitig verwenden und auf die dadurch entstehenden Vor- und Nachteile eingegangen. Außerdem erfolgt eine Analyse des Erkennungstools *Botminer*. Dieses wird in einer Testumgebung zur Erkennung des Netzwerkverkehrs eines P2P-Botnetzes verwendet. Damit wird ein Ansatz für die Analyse und Evaluierung eines Botnetz-Erkennungsprogramms präsentiert.

Ein anderer Ansatz zum Aufstellen einer Taxonomie von Erkennungsprogrammen wird in [2] gewählt. Hierbei wird zunächst zwischen der Platzierung der Programme entschieden. Dabei ist die Installation auf einem Rechner, in einem Netzwerk oder kombinierte Verfahren möglich. Bei letzterem können sowohl der Rechner, als auch das Netzwerk gleichzeitig überwacht werden. Die verschiedenen Erkennungsmechanismen werden umfangreich beschrieben und deren Vor- und Nachteile bezüglich einer Botnetz-Erkennung erläutert. Außerdem werden Methoden diskutiert, um Angriffe von Botnetzen abschwächen oder abwehren zu können. Diese Methoden werden anschließend verwendet, um die Schadsoftware des *Storm Worms* zu analysieren. Desweiteren wird die Funktionsweise des Botnetz-Erkennungsprogramms *Bothunter* vorgestellt und erläutert.

Das NIDS Snort wird in [10] von Martin Rösch analysiert. Funktionsweise und Aufbau der drei Hauptbestandteile Paket Decoder, Detection Engine und Logging/Alerting Subsystem werden ausführlich aufgezeigt. Diese Informationen können zur eigenen Analyse von Snort verwendet werden und ermöglichen ein tieferes Verständnis von dessen Arbeitsweise zur Erkennung von Schadsoftware.

In [11] werden ebenfalls die wesentlichen Komponenten des Erkennungsprogramms Snort analysiert. Hierbei werden die fünf unterschiedlichen Bestandteile Packet Decoder, Preprozessor, Detection Engine, Logging/ Alerting Subsystem und Ausgabe-Modul herausgearbeitet Diese werden ausführlich beschrieben und das Durchlaufen der einzelnen Komponenten bildlich dargestellt. Außerdem wird auf das NIDS Bro eingegangen. Auch bei diesem werden die Hauptbestandteile zur Erkennung einzeln erläutert und deren Funktion im Gesamtkontext des Programms eingeordnet. Die eigene Analyse dieser beiden Programme wird durch die ausführliche Erklärung in [11] unterstützt.

Kapitel 3

Analyse

Botnetze besitzen bei Hackern eine steigende Popularität und werden vermehrt für Angriffe verwendet. Daher ist die Erkennung von Botnetz-Infizierungen zunehmend von Bedeutung. Das Wettrüsten zwischen Botnetz-Software und Erkennungsprogrammen führt zu einer stetigen Weiterentwicklung dieser, was mit einer steigenden Komplexität der Programme einhergeht. Die Vielzahl an Software sowohl zur Erstellung, als auch zur Erkennung infiltrierter Rechner bedarf einer Einordnung nach spezifischen Merkmalen und Kriterien, da die verwendete Erkennungssoftware von der zu identifizierenden Botnetzart abhängt. [12]

Im Folgenden wird eine Kategorisierung von Botnetz-Software und Erkennungsprogrammen vorgestellt, welche eine Einordnung der Programme nach ihren unterschiedlichen Eigenschaften ermöglicht. Außerdem werden verschiedene qualitative und quantitative Kriterien zur Bewertung der Programme vorgestellt und ihre Relevanz verdeutlicht. Zusätzlich werden ausgewählte Botnetze und Erkennungsprogramme analysiert, welche gezielte Merkmale aufweisen und zur Evaluierung der definierten Kriterien verwendet werden.

3.1 Kategorisierung der Botnetz-Software

Als Unterscheidungsmerkmale zur Kategorisierung von Botnetz-Software werden folgende Punkte definiert: Netzwerk-Struktur, Kommunikationsprotokoll, Verbreitung sowie Verwendung eines Botnetz-Programms. [12, 13]

3.1.1 Netzwerk-Struktur

Unter der Netzwerk-Struktur eines Botnetzes wird die Struktur, mit der das gesamte Botnetz aufgebaut ist, verstanden und wie Befehle an die einzelnen Bots weitergegeben

werden. Dabei wird zwischen Command & Control (C&C), Peer-to-Peer (P2P) und hybriden Botnetzen unterschieden. Dies stellt eine wichtige Information für Erkennungsprogramme dar, weil sich mit der Struktur des Botnetzes auch die Vorgehensweisen ändern, mit der es erkannt oder abgeschwächt werden kann.

3.1.1.1 Command & Control

Command & Control Netzwerk-Strukturen sind hierarchisch aufgebaut und werden von den meisten, vor allem älteren Botnetzen verwendet. Im Jahr 2008 wurde der Anteil von hierarchischen Botnetzen auf etwa 95% geschätzt. [13]

Alle C&C-Botnetze besitzen einen Command & Control-Server. Eine beispielhafte Illustration eines solchen Botnetzes ist in Abbildung 3.1 zu sehen. Der Botmaster ist dabei der Besitzer des Botnetzes und kontrolliert es durch Benutzung des C&C-Servers. Sollen Angriffe auf ein bestimmtes Ziel ausgeführt werden, übermittelt der Botmaster die entsprechenden Befehle an den C&C-Server und dieser leitet sie an die Bots weiter.

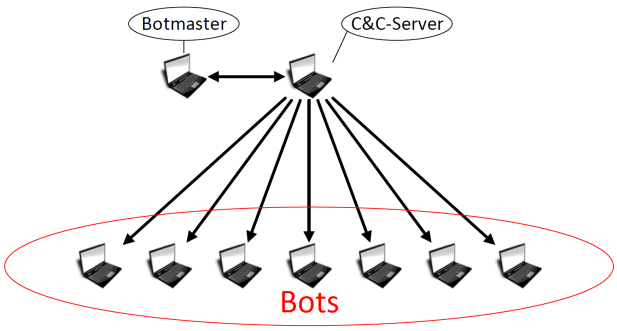

Abbildung 3.1: Aufbau eines hierarchischen C&C-Botnetzes

Durch alleinige Kommunikation mit dem C&C-Server wird die Sicherheit und Anonymität des Botmasters erhöht, da den Bots ausschließlich die IP-Adresse des C&C-Servers bekannt ist. Folglich ermöglicht eine Erkennung der Bots nur die Deaktivierung des C&C-Servers.

Für eine zusätzliche Steigerung der Anonymität des Botmasters können Bot-Controller verwendet werden. Ein Beispiel für ein solches Botnetz ist in Abbildung 3.2 dargestellt. Die Bot-Controller beteiligen sich nicht an Angriffen, sondern bilden eine Zwischeninstanz zwischen dem C&C-Server und den übrigen Bots, wodurch jegliche Kommunikation im Botnetz über sie stattfindet. Ohne Verwendung der Controller stellt die Aufdeckung des C&C-Servers eine Gefahr für den Botmaster dar, weil bei erneutem Kontaktversuch mit dem Server eine Rückverfolgung zu ihm möglich ist. Diese Gefahr wird durch Bot-Controller verringert, da erst nach deren Aufdeckung der C&C-Server ausfindig gemacht werden kann.

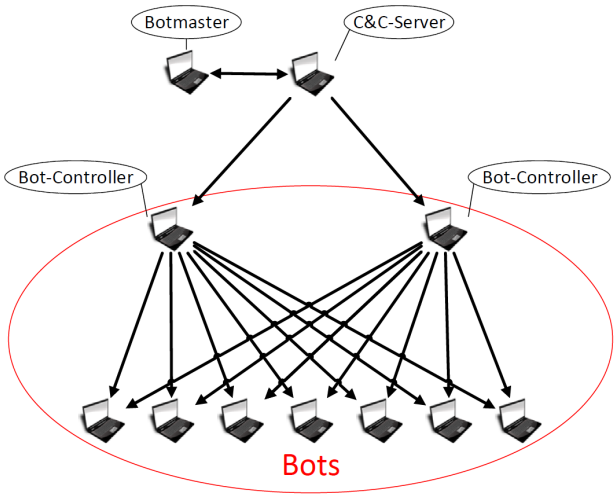

Abbildung 3.2: Aufbau eines hierarchischen C&C-Botnetzes mit Bot-Controllern

Ein Nachteil von C&C-Botnetzen ist der Verlust des gesamten Botnetzes durch die Abschaltung des C&C-Servers. Durch die Erkennung von Bots eines Botnetzes sind Nachforschungen zu der IP-Adresse des C&C-Servers möglich. Nach Aufdeckung dieser Adresse wird jegliche Kommunikation mit ihr als potenzielle Gefährdung betrachtet. Bei C&C-Botnetzen mit Bot-Controllern ist dies erschwert, da nur die Erkennung von Bot-Controllern zur IP-Adresse des C&C-Servers führt. Erkennungen der übrigen Bots enthalten nur die IP-Adressen der Bot-Controller. Werden diese erkannt, können im Botnetz neue Bot-Controller bestimmt werden. [1, 14, 15]

3.1.1.2 Peer-to-Peer

Diese Form der Netzwerk-Struktur wird auch als dezentral bezeichnet. Obwohl es sich im Jahr 2008 nur bei 5% aller Botnetze um dezentrale Netzwerk-Strukturen handelte, hat sie an Popularität und Bedeutung gewonnen. Hauptgrund dafür ist die nicht mögliche Abschaltung des Botnetzes durch Aufdeckung des C&C-Servers.

Nachfolgend ist in Abbildung 3.3 eine beispielhafte Illustration eines P2P-Botnetzes dargestellt. Hierbei existiert keine hierarchische Struktur und alle Bots sind gleichberechtigt. Durch den nicht vorhandenen C&C-Server kommunizieren die Bots ausschließlich untereinander.

Für das Hinzufügen neuer Bots zum Botnetz bestehen zwei Möglichkeiten: Entweder verwaltet jeder Bot eine Liste mit mehreren hundert Peers oder es existiert ein Server, welcher die IP-Adressen der Bots enthält und dessen eigene IP-Adresse in den Code jedes Bots integriert ist. Der Botmaster kann das Botnetz durch Kontaktierung eines beliebigen Bots kontrollieren. Übermittelt er diesem Befehle, werden sie mittels der Peer-

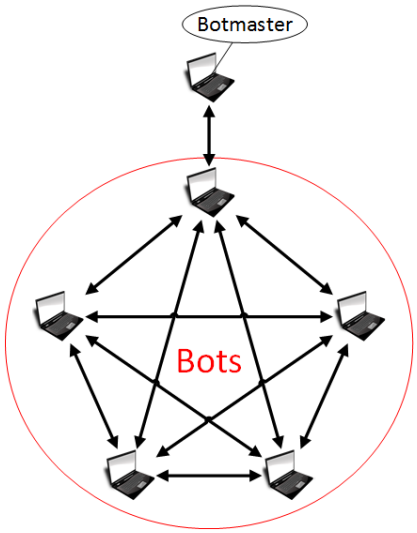

Abbildung 3.3: Aufbau eines Peer-to-Peer Botnetzes

Liste oder einem zentralem Server an alle bekannten Peers weitergeleitet. So breitet sich der Befehl im Botnetz aus, bis alle Bots Kenntnis davon haben. [14]

Anonymität für den Botmaster wird gewährleistet, indem ein beliebiger Bot kontaktiert werden kann und dieser die Verteilung im Botnetz übernimmt. Die Rückverfolgung eines DDoS-Angriffes führt dadurch nur zu den Bots. Werden für den Kontakt zum Botnetz unterschiedliche Rechner benutzt, ist eine Aufdeckung des Botmasters schwer möglich, da kein Bot für den nächsten Kontakt bestimmt werden kann. Das Aufdecken einzelner Bots beeinträchtigt die Funktion des Botnetzes kaum, da eine Kommunikation zwischen den Bots weiterhin möglich ist.

Die Verwendung von Peer-Listen führt bei der Entdeckung und Erforschung eines Bots zur Aufdeckung aller in der Liste enthaltenen Bots. Auch wenn in dieser Liste nicht alle IP-Adressen verzeichnet sind, führen die Adressen zu weiteren Bots, welche eine andere Peer-Liste besitzen und somit zur Aufdeckung des Botnetzes. Bei Verwendung eines zentralen Servers kann dessen IP-Adresse durch die Erkennung eines einzelnen Bots ermittelt werden, da jeder Bot dessen IP-Adresse besitzt. Jegliche Kommunikation im Netzwerk mit diesem Server ist dann ein Anzeichen für eine Infizierung. Außerdem erzeugen P2P-Botnetze durch regelmäßiges Austauschen von Informationen und Aktualisierungen zwischen den Bots einen großen Datenverkehr. Durch Weiterleitung von Nachrichten an alle bekannten Peers entsteht eine Vielzahl an Verbindungen mit unterschiedlichen IP-Adressen. Dieses hohe Datenaufkommen können Erkennungsprogramme zur Aufdeckung eines P2P-Botnetzes ausnutzen. [14–16]

3.1.1.3 Hybride P2P-Botnetze

Botnetze dieser Kategorie versuchen durch die gleichzeitige Benutzung einer C&C- und P2P-Struktur die Vorteile beider Botnetzarten zu vereinen. Nachfolgend ist in Abbildung 3.4 ein Beispiel für ein solches Botnetz dargestellt. Das Botnetz lässt sich in zwei Bereiche unterteilen. Der erste Bereich enthält die Servant Bots und der zweite die Client Bots. Erstere verfügen über eine global erreichbare IP-Adresse und können sowohl als Clients, als auch als Server fungieren. Clients akzeptieren keine eingehenden Verbindungsanfragen. Zu dieser Gruppe zählen alle Bots, welche über eine dynamisch zugewiesene IP-Adresse, eine private IP-Adrese oder eine durch eine Firewall geschützte IP-Adresse besitzen. Die Servant Bots sind in einem P2P-Netzwerk miteinander verbunden und enthalten in ihren Peer-Listen die IP-Adressen von benachbarten Servant Bots. Die Client Bots können nur mit einem festgelegten Servant Bot kommunizieren. Der Botmaster kann zur Kommunikation einen beliebigen Bot wählen. Kontaktiert er einen Servant Bot, wird die Nachricht an die Nachbarn weitergeleitet, bis sie alle Servant Bots erreicht hat. Bei Übermittlung der Befehle an einen Client Bot werden sie an den zugewiesenen Servant Bot weitergeleitet.

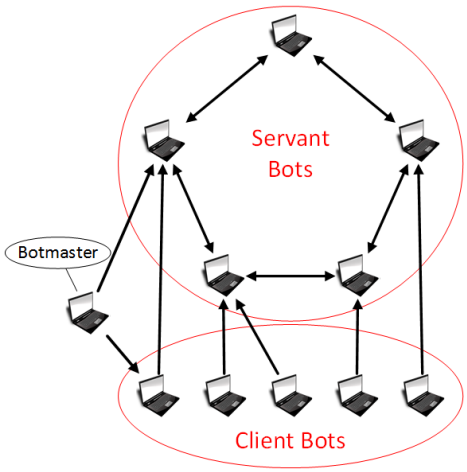

Abbildung 3.4: Aufbau eines hybriden Peer-to-Peer Botnetzes

Ein Vorteil hybrider P2P-Botnetze ist die Berücksichtigung von privaten, automatisch generierten und durch Firewall geschützte IP-Adressen. Da Geräte mit solchen Adressen nicht vom Internet aus erreichbar sind, wird ihnen keine essenzielle Aufgabe im Botnetz übertragen. Somit gewinnt das Botnetz an Robustheit. Außerdem sind hybride P2P-Botnetze schwerer abzuschalten, da die Entdeckung eines Client Bots lediglich Rückschlüsse zu einem einzigen Servant Bot zulässt. Bei Erkennung eines Servant Bots enthält die zugehörige Peer-Liste nur die IP-Adressen von benachbarten Bots.

Ein Nachteil von hybriden P2P-Botnetzen ist die notwendige Gewährleistung einer ausreichenden Anzahl an Servant Bots, um ein stabiles Backbone bilden zu können.

Sind dafür nicht genügend Bots verfügbar, ist das Netzwerk mit einem C&C-Botnetz mit Bot-Controllern vergleichbar und besitzt ähnliche Schwächen. Außerdem erzeugen die Servant Bots einen auffällig hohen Datenverkehr und können zur Entdeckung des Botnetzes führen. [14, 16, 17]

3.1.2 Kommunikationsprotokoll

Da jedes Botnetz ein Kommunikationsprotokoll zur Kontrolle der Bots benötigt stellt dies für Erkennungsprogramme eine Möglichkeit zur Aufdeckung dieser dar. Neben einem Botnetz kommunizieren viele andere Anwendungen auf einem Rechner. Dadurch ist die genaue Kenntnis von Botnetz-Kommunikationsprotokollen notwendig, um diese zur Botnetz-Erkennung verwenden zu können. Die am weitesten verbreiteten Protokolle sind IRC und HTTP. Diese werden nur von hierarchischen Botnetzen verwendet. P2P-Botnetze benutzen spezielle P2P-Protokolle wie zum Beispiel das Overnet-Protokoll, welche sich untereinander unterscheiden. Dadurch ist bei ihnen kein einheitliches Kommunikationsprotokoll definierbar. [18, 19]

3.1.2.1 IRC

Der Internet-Relay-Chat (IRC) wurde ursprünglich für die Client-Server-Kommunikation entwickelt. Dabei können sich Clients mit Hilfe eines eindeutigen *nicknames* zu einem Server verbinden. Außerdem existieren verschiedene Kanäle, in die sich Nutzer einloggen können. [20]

Dieses Protokoll wurde aus verschiedenen Gründen für die Kommunikation von Botnetzen verwendet. Ein Aspekt ist die Möglichkeit, dass in einem Channel mehrere hundert Rechner miteinander kommunizieren können und somit für große Botnetze eine genügend große Kapazität vorhanden ist. Auch ermöglicht der sofortige Nachrichtenaustausch eine schnelle Interaktion zwischen Botmaster und Bots. Desweiteren existieren verschiedene Open-Source Implementierungen, wodurch die generelle Flexibilität des IRC-Protokolls unterstützt und die Anpassung an die individuellen Anforderungen verschiedener Botnetze ermöglicht wird.

Für ein Botnetz wird ein Channel erzeugt, über den die anschließende Kommunikation zwischen Botmaster und Bots stattfindet. Nach Ausführung der Botnetz-Schadsoftware auf einem Rechner versucht dieser, Kontakt zu seinem IRC-Server herzustellen. Dies ist entweder durch eine direkt angegebene IP-Adresse im Botnetz-Code oder durch eine Namensauflösung mittels DNS möglich. Sobald dem Bot die Adresse des Servers bekannt ist, versucht er den Aufbau einer IRC-Verbindung zu diesem und den Beitritt zu einem in den Binärdaten definierten Channel. Sowohl für die Verbindung zum Server, als auch zum Channel ist eine Authentifikation erforderlich. Die Befehle des Botmasters können unmittelbar nach dem Verbindungsaufbau mit dem IRC-Server sowie dem

Eintritt in den gemeinsam genutzten Channel ausgeführt werden. Sofern keine weiteren auszuführenden Anweisungen vorliegen, verbleibt der Bot im IRC-Channel und wartet auf Befehle des Botmasters.

Nachteil von IRC-Botnetzen ist die Auffälligkeit des Netzwerkverkehrs, da es in der heutigen Zeit als Chat-Programm an Popularität verloren hat. Dadurch führt ein vorliegender Datenverkehr mit diesem Protokoll, obwohl die Benutzer des Netzwerks derartige Chatprogramme nicht verwenden, zu einer hohen Wahrscheinlichkeit für eine Botnetz-Infizierung und kann zur Erkennung der Bots verwendet werden. Dies führt zu einer zunehmenden Nutzung von HTTP-Kommunikationsprotokollen. [20, 21]

3.1.2.2 HTTP

Das Hypertext Transfer Protocol (HTTP) ist ein Protokoll zur Übertragung von Daten und arbeitet nach dem ISO/OSI-Modell auf der Anwendungsschicht. Der Hauptverwendungszweck dieses Protokolls ist das Laden von Webseiten aus dem World Wide Web (WWW) zur Darstellung in Browsern. Auch nicht WWW-basierte Übertragungen sind mit HTTP möglich.

Bei Betreibern von Botnetzen gewinnt HTTP zunehmend an Popularität. Da die meisten Botnetze das IRC-Protokoll zur Kommunikation benutzen und auftretender IRC-Datenverkehr oftmals auffällig ist, senkt die Verwendung des HTTP-Protokolls das Entdeckungsrisiko. Die Botnetz-Erkennung mittels HTTP-Datenverkehr wird durch das Aufbauen zahlreicher HTTP-Verbindungen zu verschiedenen Webseiten durch den Nutzer eines Computers erschwert. Dadurch ist die Unterscheidung zwischen legitimem und Botnetz-Datenverkehr schwer möglich. Außerdem wird ausgehender HTTP-Datenverkehr meist nicht von Firewalls blockiert.

Bei Verwendung von HTTP für Botnetze wird die IP-Adresse oder URL des Webservers, welcher als Command & Control-Server fungiert, in die Binärdaten des Botnetz-Schadcodes integriert. Nach der Infizierung eines neuen Rechners kann der entsprechende Webserver aufgerufen und bezüglich auszuführender Befehle durchsucht werdem. In regelmäßigen Abständen besucht ein Bot den Webserver und wartet auf eintreffende Befehle. Der Botmaster kann das Botnetz steuern, indem er auszuführende Aktionen auf dem Webserver veröffentlicht. Sobald die Bots das nächste mal den Server besuchen, laden sie die Befehle herunter und führen sie aus. [18]

3.1.3 Verbreitung

Zur Etablierung eines Botnetzes ist die Verbreitung der Schadsoftware erforderlich. Weiterverbreitungen sind dabei über E-Mails, infizierte Links, Webseiten oder durch das aktive Scannen nach Schwachstellen möglich und stellen eine Möglichkeit zur

Entdeckung für Erkennungsprogramme dar, bevor die Schadsoftware ausgeführt wird. Sobald durch versehentliches Anklicken eines Links die Schadsoftware heruntergeladen wird, installiert sich diese automatisch und der Bot kann das Scannen seiner Umgebung beginnen.

Beim Scan-Verhalten sind Botnetze in zwei Kategorien unterteilbar: Zur ersten gehören Wurm ähnliche Botnetze, welche kontinuierlich nach bestimmten Ports zur Weiterverbreitung scannen. Zur zweiten Kategorie sind Botnetze mit einem variablen Scan-Verhalten zu zählen. Diese besitzen verschiedene Scan-Algorithmen, die sie an ihre Umgebung anpassen können und scannen nur bei Erhalt eines entsprechenden Befehls. Durch ihr unvorhersehbares Verhalten sind sie für Erkennungsprogramme schwerer aufzudecken. [13, 21]

3.1.4 Verwendung

Eine weitere Möglichkeit, zwischen verschiedenen Botnetzen zu differenzieren ist ihre Verwendung. Werden Bots für einen spezifischen Zweck verwendet, müssen sie dafür zum Beispiel durch das Senden von Paketen aktiv werden. Verschiedene Verwendungen erzeugen dabei unterschiedliche Bot-Aktivitäten welche zur Botnetz-Erkennung verwendet werden können.

DDoS-Angriff: Dies sind Angriffe, welche das Ziel haben, ein Opfer zu überlasten, indem eine Vielzahl an Anfragen von den Bots an eine Webseite oder einen Server geschickt werden. Dadurch ist die Beantwortung legitimer Anfragen nicht mehr möglich und führt zu deren Verwerfung. Als Ergebnis sind betroffene Webseiten für legitime Nutzer nicht mehr erreichbar. DDoS-Angriffe sind Hauptverwendungszweck von Botnetzen und werden sowohl für finanzielle, als auch politische Gründe genutzt. [15]

Spamming: Im Jahr 2008 wurden 70 bis 90 % der weltweiten Spam-Pakete durch Botnetze verursacht. Bots können zunächst zum Sammeln von E-Mail-Adressen verwendet werden, indem sie auf einem infizierten Rechner Benutzereingaben aufzeichnen und Informationen über geschriebene E-Mails sammeln. Die zugehörigen Empfänger werden gespeichert und an den Botmaster weitergeleitet. Dieser verwendet die Bots anschließend, um an alle gesammelten Email-Adressen Spam-Mails zu verschicken. Außerdem können die gesammelten Adressen zur Verbreitung des Botnetz-Schadcodes genutzt werden, weil bei einem Versenden infizierter Links das Anklicken dieser zum Herunterladen des Schadcodes auf den Rechner führt. Auf diesem kann sich die Schadsoftware automatisch installieren, unbemerkt Daten sammeln und auf Befehle des Botmasters warten. [15]

Informationsdiebstabl: Sind Bots unbemerkt auf fremden Rechnern installiert, können sie sämtliche Benutzereingaben aufzeichnen. Dadurch gelangen sie in den Besitz sensibler Informationen wie Benutzernamen, Passwörter, Bank- oder Kreditkartendaten.

Diese können für einen Angriff auf das Opfer benutzt werden. [15]

Klickbetrug: Mit der Hilfe von Botnetzen ist die Installation von Werbe Add-ons und Browser Helper Objects (BHO) für wirtschaftliche Zwecke möglich. Durch ein regelmäßiges Aufrufen spezieller Hyperlinks können sie zum Erreichen höherer Klickraten benutzt werden und somit für ein Produkt werben. [15]

3.2 Vorstellung verwendeter Botnetz-Software

Im Folgenden werden die verwendeten Programme vorgestellt, welche zur Überprüfung verschiedener Erkennungskriterien verwendet werden. Die verwendeten Botnetze sind BeEF und Zeus.

3.2.1 BeEF

BeEF steht für Browser Exploitation Framework und bezeichnet ein Sicherheitsprogramm, welches für Penetrationstests verwendet werden kann. Der Fokus liegt dabei auf Webbrowsern. Ursprünglich wurde es zur Demonstration von Cross-Site-Scripting verwendet und hat sich durch Bereitstellung verschiedener Client-seitiger Angriffsbeispiele zu einem populären Tool für Tester entwickelt. BeEF ist außerdem ein frei erhältliches Open-Source-Programm, was nicht nur bei Testern, sondern auch bei Hackern zu einer großen Beliebtheit geführt hat.

BeEF wurde als zu evaluierendes Botnetz ausgewählt, da es sich bei seiner Netzwerk-Struktur um ein C&C-Botnetz ohne Bot-Controller handelt und es somit bezüglich dieser Eigenschaft für den überwiegenden Anteil an Botnetzen repräsentativ ist. Außerdem verwendet BeEF das weit verbreitete HTTP-Protokoll zur Kommunikation. Dadurch wird eine Erkennung aus den in Kapitel 3.1.2.2 angegebenen Gründen erschwert. Zusätzlich kann ein beliebiger Port für die Kommunikation mit dem C&C-Server gewählt werden. Dadurch wird eine Erkennung wegen der Benutzung eines ungewöhnlichen Ports verhindert. Die Verwendung eines Proxy-Servers ist möglich, um die Anonymität des Botmasters zu erhöhen.

BeEF bezeichnet die Infizierung von Bots unabhängig von der verwendeten Methode als Hooking. Es besitzt keine selbstständigen Weiterverbreitungsmöglichkeiten, sondern infiziert neue Rechner, indem diese eine spezielle Webseite aufrufen, welche das Herunterladen eines Javaskripts vom BeEF C&C-Server auslöst. Dieses Skript wird daraufhin in regelmäßigen Abständen vom Bot angefordert. Soll ein Kommando an einen bestimmten Bot gesendet werden, wird dies über den C&C-Server ausgelöst. Beim nächsten Anfordern des Javaskripts dieses Bots, werden die Befehle mit dem Skript übermittelt und der Bot beginnt die Ausführung. Neben der Erstellung eigener Websei-

ten zum Hooken neuer Browser können auch von BeEF bereitgestellte Demowebseiten verwendet werden.

Bei Beef handelt es sich um ein modulares Framework. Es besitzt eine große Anzahl vorinstallierter Kommandos und ermöglicht durch Bereitstellung einer eigenen API die Entwicklung eigener Module. Mit den verfügbaren Kommandos ist eine Vielzahl verschiedener Verwendungszwecke wie das Scannen des Bots oder DDoS-Angriffe möglich. [7, 22]

3.2.2 Zeus

Zeus wurde in einer Sicherheitsveröffentlichung im Jahr 2009 als Botnetz mit der größten Gefährdung festgesetzt. Mit einer Größe von geschätzten 3,6 Millionen infizierten Geräte soll Zeus für 44% der Infizierungen von Banken verantwortlich gewesen sein. Gründe für die Popularität von Zeus waren unter anderem sein freundliches Benutzer-Interface und ein niedriger Preis auf dem Schwarzmarkt.

Zeus wurde zur Evaluation ausgewählt, da es sich um das populärste Botnetz handelt und für die Infizierung einer großen Anzahl an Rechnern verantwortlich war. Dadurch muss es Mechanismen besitzen, um eine Erkennung zu erschweren beziehungsweise zu verhindern. Durch seine Verwendung sollen aktuelle Erkennungsprogramme überprüft werden, ob mit ihnen inzwischen die Erkennung dieses Botnetzes möglich ist.

In den Tests wurde die Zeus Version 2.0.8.9 verwendet, bei welcher es sich um die aktuellste frei zugängliche Version handelt. Zeus verwendet einen Command & Control-Server, welcher mittels diversen PHP-Skripten die Steuerung und Kontrolle des Botnetzes übernimmt. Desweiteren verwendet es für eine erschwerte Erkennung das HTTP-Protokoll und verschlüsselt die Kommunikation des gesamten Botnetzes.

Zur Infizierung neuer Rechner nutzt Zeus eine ausführbare Bot-Datei. Diese können über Spam-Mails oder spezielle Webseiten auf einen Rechner heruntergeladen werden. Sobald die Datei vom Benutzer angeklickt wird, kopiert sie sich in einen System-Ordner des Betriebssystems und fordert mittels dem Befehl *GET /config.bin* vom Zeus C&C-Server die verschlüsselte binäre Konfigurationsdatei an. Diese wird vom Bot mittels dem im Bot-Code integrierten Key entschlüsselt und enthält Informationen über die Konfiguration des Bots. Entsprechende Befehle werden über die binäre Konfigurationsdatei weitergegeben und in Abständen von circa einer Minute von den Bots kontinuierlich angefordert. Gesammelte Daten werden von den Bots mit dem Befehl *POST /gate.php* an der C&C-Server geschickt.

Zeus wird hauptsächlich zur Spionage von Bankdaten, Kreditkartendaten oder anderen sensiblen Nutzerdaten verwendet. Grund dafür ist das benutzerfreundliche Interface zur Bedienung und die standardmäßige Integration von Keyloggern, welche von den Bots zur Aufzeichnung der Benutzerdaten verwendet werden. [8]

3.3 Evaluationskriterien für Botnetz-Software

Zur Bewertung unterschiedlicher Botnetz-Software ist die Definition vergleichbarer Kriterien notwendig. Hierfür wurden sowohl qualitative, als auch quantitative Kriterien gewählt, um möglichst viele verschiedene Aspekte abzudecken. Es wurden folgende Kriterien festgesetzt: Effektivität, Verborgenheit, Weiterverbreitung, Ausfallsicherheit und Portabilität.

3.3.1 Funktionsumfang

Der Funktionsumfang ist ein Kriterium zur Bewertung von Botnetzen bezüglich der bereitgestellten Funktionalitäten. Hierbei wird betrachtet, wie viele verschiedene Funktionen das Botnetz anbietet und ob die Funktionen ihre Aufgabe erfüllen beziehungsweise das angestrebte Ergebnis erreicht wird. Außerdem wird bewertet, ob die Botnetze eine Möglichkeit zur Erweiterung durch das Definieren eigener Funktionen oder Module besitzen und inwiefern dies beispielweise durch das Anbieten einer API unterstützt wird.

3.3.2 Verborgenheit

Das Kriterium der Verborgenheit dient zur Bewertung der Fähigkeiten des Botnetzes, nicht von Benutzern oder Erkennungsprogrammen aufgedeckt zu werden. Dies wird durch die zwei Faktoren Verwendungseffizienz und Dateneffizienz sowie durch die Anstrengungen des Botnetzes bestimmt, welche dieses unternimmt, um eine Erkennung zu vermeiden. Diese Anstrengungen können sich dabei unter anderem auf das Verwenden unauffälliger Prozessnamen oder Speicherorte beziehen.

Verwendungseffizienz: Dies bezeichnet die Effizienz des Botnetzes bei der Ausführung verschiedener Aktionen. Werden dabei weniger Ressourcen benötigt, verringert dies die Wahrscheinlichkeit einer Erkennung, da eine geringere Anomalie erzeugt wird. Dies wird begünstigt, wenn dem Botmaster eine größere Anzahl an Bots oder Bandbreite zur Verfügung steht, da beispielweise die Durchführung eines DDoS-Angriffs zu einer geringeren Auslastung der System- und Netzwerkressourcen eines Bots bei gleicher Auswirkung beim Opfer führt.

Dateneffizienz: Die Dateneffizienz bezeichnet die Menge und Größe der ausgetauschten Nachrichten im Botnetz. Ein erhöhter Datenverkehr führt zu einer zunehmenden Auffälligkeit im Netzwerk und erhöht das Risiko einer Entdeckung. Wesentlichen Einfluss darauf haben das Update-Intervall der Bots und die benötigte Zeitspanne für einen Nachrichtenaustausch. Ein kürzeres Update-Intervall der Bots untereinander oder zwischen Bots und C&C-Server erhöht die Anzahl ausgetauschter Nachrichten und steigert

somit das Risiko einer Entdeckung. Am unauffälligsten ist dabei ein unregelmäßiges Intervall, bei dem Bots nur zum Senden oder Empfangen neuer Informationen Nachrichten austauschen, da dies kein regelmäßiges Muster erzeugt. Ein schnellerer Nachrichtenaustausch verkürzt die Zeitspanne einer möglichen Erkennung. Außerdem können dadurch zu viele Pakete in kurzer Zeit auftreffen, wodurch nicht alle vom Erkennungsprogramm analysiert werden können.

3.3.3 Weiterverbreitung

Das Kriterium der Weiterverbreitung bewertet Botnetze bezüglich ihren Möglichkeiten zur Rekrutierung neuer Bots. Dabei kann differenziert werden zwischen Botnetzen, die selbstständige Weiterverbreitungsroutinen besitzen und Botnetzen, welche manuell weiterverbreitet werden. Bei Ersterem Scannen die Bots ihre Umgebung aktiv nach Schwachstellen und nutzen sie für eine Infizierung aus. Dadurch können weitere Rechner infiziert werden und die Größe des Botnetzes nimmt zu. Diese schnelle Weiterverbreitung und erhöhte Aktivität der Bots steigert jedoch die Gefahr einer Erkennung des Botnetzes. Können sich Bots nicht selbstständig weiterverbreiten, kann dies stattdessen über das Verschicken infizierter E-Mails oder Webseiten stattfinden. Dies führt zu einem langsameren Wachstum des Botnetzes und reduziert aufgrund einer geringeren Aktivität der Bots die Gefahr einer Erkennung. Soll ein Botnetz langfristig bestehen, ist daher eine manuelle Verbreitung von Vorteil.

3.3.4 Ausfallsicherheit

Mit der Ausfallsicherheit wird ein Botnetz bezüglich seiner Fähigkeit bewertet, mit ausfallenden Bots umzugehen. Dies ist unter anderem relevant, da es die Auswirkungen aufzeigt, welche die Abschaltung einzelner Bots zur Folge hat.

Eine höhere Anzahl an Verbindungen zwischen den Bots begünstigt die Ausfallsicherheit eines Botnetzes, da der Verlust eines Bots die Kommunikation mit anderen nicht beeinträchtigt. Dies kann durch die lokale Transitivität bestimmt werden, welche zur Veranschaulichung in Abbildung 3.5 dargestellt ist. Es wird zwischen zwei Fällen unterschieden und die Knoten repräsentieren Bots.

- Fall 1: Bot C ist von A direkt und zusätzlich über B erreichbar. Fällt Bot B aus, kann mit C weiterhin über A kommuniziert werden.
- Fall 2: Mit Bot C kann von A nur über B kommuniziert werden. Fällt Bot B aus, ist C nicht mehr erreichbar.

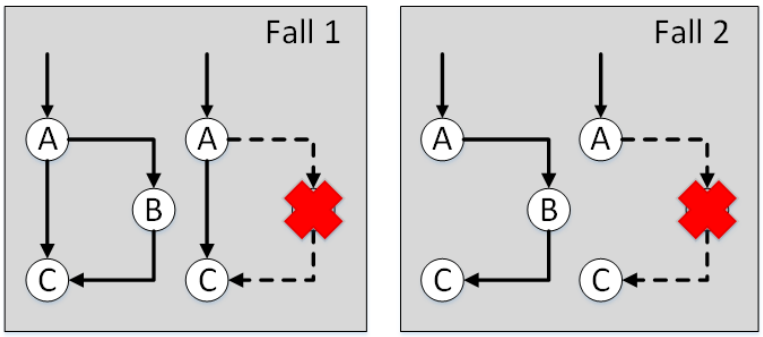

Abbildung 3.5: Beschreibung der Transitivität

3.3.5 Portabilität

Die Portabilität bewertet ein Botnetz bezüglich seiner Kompatibilität mit unterschiedlichen Umgebungsbedingungen. Sind Botnetze nicht auf die Infizierung spezifischer Betriebssysteme, Architekturen oder Programme beschränkt, können mehr Rechner infiziert werden und das Botnetz besitzt eine höhere Portabilität.

3.3.6 Anonymität

Dieses Kriterium beschreibt die Schwierigkeit, mit der ein Botmaster aufgedeckt werden kann. Wichtigste Einflussgröße auf die Anonymität stellt die Netzwerk-Struktur dar. Wie bereits besprochen unterscheiden sich C&C-Botnetze und P2P-Botnetze in der für den Botmaster gewährten Anonymität. Außerdem kann die Anonymität durch Verwendung eines Proxy-Servers erhöht werden. Dabei werden alle Aktionen des Botmasters über diesen Server weitergeleitet, wodurch eine Rückverfolgung von Angriffen zum Botmaster zusätzlich erschwert wird. [1]

3.3.7 Benutzbarkeit

Die Bewertung dieses Kriteriums inkludiert vorhandene Dokumentationen über das Programm, die Schwierigkeit der Installation und die Benutzerfreundlichkeit des Interfaces zur Steuerung des Botnetzes. Da diese Eigenschaften wissenschaftlich nicht nachgewiesen werden können, werden diverse Beobachtungen und Erfahrungen beschrieben, die während der Integration in das Testbed festgestellt werden konnten. Dabei wird im Speziellen auf die Command & Control-Server der verwendeten Botnetze eingegangen.

3.4 Kategorisierung der Erkennungssoftware

Da unterschiedliche Botnetze durch verschiedene Erkennungsprogramme aufgedeckt werden können, ist die Definition verschiedener Verfahren für die Bewertung notwendig. Im Folgenden werden Erkennungsprogramme nach den Verfahren mittels Signatur, Anomalie und kombinierte Verfahren unterteilt. Außerdem werden verschiedene Platzierungen der Erkennungsprogramme vorgestellt.

3.4.1 Signatur

Signatur-basierte Ansätze zur Erkennung von Botnetzen versuchen Malware anhand seiner Charakteristika oder Signatur zu erkennen. Dabei kann zusätzlich zwischen Host- und Netzwerk-basierter Signatur-Erkennung unterschieden werden.

Ein Vorteil von Signatur-basierten Verfahren ist eine niedrige Falsch-Positiv-Rate. Indem bekannte Signaturen oder Charakteristika verwendet werden, um mögliche Schadsoftware zu erkennen, besteht dabei eine geringe Fehlerquote, da nur geringfügige Abweichungen des Merkmals eine Erkennung verhindern.

Nachteil von Erkennungen mittels Signatur ist, dass nur bekannte Schadsoftware identifizierbar ist und somit kein Schutz vor neuen oder unbekannten Botnetzen besteht. Diese können erst nach Aufdeckung und Erforschung mehrerer Bots erkannt werden, wenn spezifische Signaturen festgestellt werden konnten. Dadurch können keine Botnetze erkannt werden, welche keine eindeutige Signatur besitzen oder diese ändern. Desweiteren muss jeder Rechner eine sichere Liste über bekannte Signaturen speichern und diese regelmäßig updaten. Werden von den Benutzern keine regelmäßigen Updates durchgeführt, ist kein zuverlässiger Schutz mehr gewährleistet. [1, 9]

3.4.1.1 Host-basierte Signatur-Erkennung

Bei dieser Form der Signatur-Erkennung steht die Überprüfung und Überwachung von internen Zuständen des Rechners, Systemressourcen und Systemverhalten im Vordergrund. Werden hierbei spezifische Charakteristika oder Signaturen entdeckt, wird der Benutzer des Rechners benachrichtigt oder der entdeckte Bot-Code automatisch entfernt.

Durch ein regelmäßiges Scannen des Computers nach bekannten Signaturen und Charakteristika wird auch dauerhaft ein zuverlässiger Schutz gegen Botnetze mit bekannter Signatur gewährleistet. Spezifisch für diese Art der Botnetzerkennung sind Antivirenprogramme. [2]

3.4.1.2 Netzwerk-basierte Signatur-Erkennung

Bei dieser Form der Signatur-Erkennung wird der Netzwerkverkehr hinsichtlich bekannter Signaturen und Charakteristika überwacht. Bei solchen Merkmalen kann es sich um auffällige IP-Adressen, Ports, verwendete Dienste oder Kommunikationsprotokolle handeln.

Bei populären Botnetzen wie dem Zeus Botnetz existieren im Internet öffentliche IP-Blocklists, welche IP-Adressen und Domain Namen von bekannten Zeus Command-&Control-Servern enthält. Durch Herunterladen dieser Blocklists und anschließendem Vergleich mit kommunizierten IP-Adressen im Netzwerk ist eine Erkennung von Botnetz-Datenverkehr möglich. [23]

Desweiteren kommunizieren vor allem ältere Botnetze über untypische Ports mit ihren Bots. Da dies zum Beispiel beim Botnetz Zeus ein Bereich zwischen 1024 und 30000 ist, stellt die Erkennung einer Kommunikation über einen solchen Port eine Unauffälligkeit im Netzwerkverkehr dar. Da Botnetze ihren verwendeten Port variieren können, ist eine genaue Identifizierung einer Infizierung jedoch nicht möglich.

Botnetze verwenden verschiedene Kommunikationsprotokolle wie IRC oder HTTP, um miteinander zu kommunizieren. Wird Datenverkehr über ein solches Protokoll entdeckt, kann dies zwar eine Auffälligkeit darstellen, eine eindeutige Zuordnung zu einem Botnetz ist aber meist nicht möglich. [1, 9]

3.4.2 Anomalie

Anomalie-basierte Erkennungsprogramme erkennen Schadsoftware durch die Feststellung eines ungewöhnlichen Verhaltens. Da dieses Verhalten sowohl den Host-Rechner, als auch das Netzwerk betreffen kann, wird zwischen Host-basierter Anomalie-Erkennung und Netzwerk-basierter Anomalie-Erkennung differenziert.

Ein großer Vorteil von Anomalie-basierten Erkennungsprogrammen ist die Möglichkeit zur Erkennung von auch unbekannten Botnetzen, da ausschließlich ein untypisches Verhalten überprüft wird.

Ein Nachteil dieser Erkennungsprogramme ist ihre hohe Falsch-Positiv-Rate. Jegliche Abweichung vom Normalverhalten verursacht die Entdeckung von Schadsoftware. Unregelmäßigkeiten sind allerdings nicht immer durch ein Botnetz begründet, sondern können auch durch natürliche Veränderungen der Umwelt entstehen. [9]

3.4.2.1 Host-basierte Anomalie-Erkennung

Host-basierte Anomalie Erkennungen überprüfen die Zustände, Systemressourcen und das Systemverhalten des Rechners auf Unregelmäßigkeiten. Durch regelmäßiges Scannen des Computers ist stets ein aktueller Bericht des Normalververhaltens des Rechners vorhanden. So können geringe Abweichungen entdeckt und gemeldet werden.

Ist ein Rechner mit einer Botnetz-Software infiziert, kann dies beispielsweise Auswirkungen auf dessen Systemressourcen haben. Da Bots ihre Umgebung nach Schwachstellen untersuchen, benötigt dies entsprechende Rechenleistung. Somit kann eine erhöhte CPU-Auslastung, obwohl kein bekanntes Programm dafür verantwortlich ist, Rückschlüsse auf eine mögliche Infizierung liefern. Allerdings ist diese Nachweismethode meist ungenau, weil in einem Computersystem zu viele Programme durchgehend aktiv sind, um eine erhöhte CPU-Auslastung korrekt zuweisen zu können. [2]

3.4.2.2 Netzwerk-basierte Anomalie-Erkennung

Hierbei werden Botnetze anhand verschiedener Netzwerk-Anomalien erkannt. Bei diesen Anomalien kann es sich um eine hohe Netzwerk-Latenz, große Datenmengen, die Benutzung ungewöhnlicher Ports oder die Kommunikation mit ungewöhnlichen IP-Adressen handeln.

Sind Rechner in einem Netzwerk mit einem Botnetz infiziert, kann dies große Datenmengen zur Folge haben. Bei der Ausführung eines DDoS-Angriffes senden alle daran beteiligten Bots mit ihrer maximalen Netzwerkkapazität Pakete an das zu überlastende Opfer. Eine solche Beteiligung des Netzwerkes weist also eine außergewöhnlich hohe Datenmenge auf und führt zu einer starken Anomalie, welche zur Erkennung des Botnetzes genutzt werden kann.

Sind mehrere Rechner in einem Netzwerk Teil eines Botnetzes, kann der Netzwerkverkehr bezüglich gleichen Kommunikationsmustern überprüft werden. Bei hierarchischen Botnetzen kommunizieren die Bots meist in gleichen Abständen mit ihrem C&C-Server. Tritt also eine regelmäßige Kommunikation zwischen mehreren Rechnern im Netzwerk und einer ungewöhnlichen IP-Adresse auf, stellt dies ein abnormales Verhalten dar und wird vom Erkennungsprogramm entdeckt. Auch ist das Kommunizieren von mehreren Rechnern mit einer untypischen IP-Adresse und Port auffällig und erlaubt Rückschlüsse auf eine Botnetz. [2, 9]

3.4.3 Kombinierte Verfahren

In der Realität sind aktuelle Botnetz-Erkennungsprogramme meist nicht nur einem Bereich der Signatur- oder Anomalie-Erkennung zuzuordnen. Um die Vorteile aus meh-

reren Bereichen zu nutzen und damit die Nachteile zu verringern, werden übergreifende Verfahren angewandt, um einen besseren Schutz vor Botnetzen zu gewährleisten.

Mining-basierte Ansätze sind die umfassendsten Erkennungsprogramme für Botnetze. Durch Überwachung und Aufzeichnung von Host-basierten und Netzwerk-basierten Aktivitäten sollen die Vorteile der verschiedenen Verfahren kombiniert werden. Hierfür werden alle verfügbaren Log-Datein und miteinander in Beziehung stehende Events überprüft. [2, 9]

3.4.4 Platzierungen

Ein anderer Ansatz zur Kategorisierung von Erkennungsprogrammen ist die Unterscheidung bezüglich ihrer Platzierung. Dabei kann zwischen Host Intrusion Detection Systems (HIDS) und Network Intrusion Detection Systems (NIDS) unterschieden werden. Erstere verwenden Host-basierte Verfahren und zweitere Netzwerk-basierte Verfahren zur Erkennung von Botnetzen. Eine Unterscheidung der Erkennungsprogramme im Bezug auf die Platzierung hat den Vorteil, dass nur unterschieden werden muss, ob die Software auf einem zu überwachenden Rechner oder in einem zu prüfenden Netzwerk installiert werden muss. Host-basierte Programme müssen auf jedem zu prüfenden Computer im Netzwerk integriert werden. Netzwerk-basierte Programme dagegen werden nur einmal auf einem zentralen Netzwerkknoten installiert, um den Datenverkehr von allen Rechnern im Netzwerk zu überwachen. [24]

3.5 Vorstellung verwendeter Erkennungstools

Basierend auf den Unterscheidungskriterien von Erkennungsprogrammen wurden Programme ausgewählt, die sich aufgrund ihrer Eigenschaften zur Evaluation und Erkennung von Botnetzen besonders eignen. Bei diesen handelt es sich um Snort, Bro und Antidoto.

3.5.1 Snort

Bei Snort handelt es sich um ein frei erhältliches Network Intrusion Detection System (NIDS). Es wurde ausgewählt, da es sich um das populärste NIDS handelt und eine Erkennung von auch unbekannten Botnetzen ermöglicht.

Snort kann zur Echtzeit-Analyse mittels libpcap basierte Netzwerk Sniffer oder zum Loggen von Datenverkehr verwendet werden. Beim Paketloggen werden integrierte Regeln verwendet, wodurch eine Erkennung von Schadsoftware durch auftretende Muster ermöglicht wird. Desweiteren besitzt Snort eine Echtzeit Alarm-Funktion. Meldungen

werden dabei an das Syslog, Server Message Block (SMB) Popup-Meldungen oder an eine spezielle alert-Datei gesendet und können dort vom Benutzer eingesehen werden.

Zahlreiche Regeln zur Datenanalyse sind in Snort bereits standardmäßig integriert. Speziell für Botnetze ist ein eigener Regelsatz erhältlich. Weitere Regeln der öffentlichen Gemeinschaft von Snort sind über [25] verfügbar. Eigene Regeln können basierend auf der spezifischen Beischreibungssprache dafür oder durch Schreiben mittels der Programmiersprache C hinzugefügt werden. Die Effektivität von Snort ist von der Qualität der definierten Regeln abhängig. Regeln, welche zu viele Pakete betreffen, können zu einer hohen Falsch-Positiv-Rate führen und somit eine Analyse erschweren. Zu genau definierte Regeln werden nicht sämtliche Formen der Schadsoftware erkennen können und die Falsch-Negativ-Rate erhöhen.

Da sich Signaturen von Botnetzen ändern, ist ein kontinuierliches Anpassen und Aktualisieren der Regeln erforderlich. Für eine zuverlässige Erkennung benötigt Snort Zugriff auf alle Daten, um die Erstellung von Datenströmen zu ermöglichen und somit eine Erkennung zu verbessern. Außerdem ist es mit Snort möglich, den Payload von Paketen zu inspizieren. Somit ist eine regelbasierte Analyse der Pakete bezüglich der in der Anwendungsschicht enthaltenen Daten möglich und erlaubt Snort die Erkennung verschiedener Schadsoftware. Außerdem sind Filterfunktionen verfügbar, um die Anzeige von gezielten Pakettypen zu ermöglichen.

Snort verwaltet die integrierten Regeln in doppelt verketteten Listen, welche als Chain Headers und Chain Options bezeichnet werden. Dabei handelt es sich um Regellisten, bei denen gemeinsame Attribute in den Chain Headers und bestimmende Erkennungsoptionen in den Chain Options gespeichert werden. Diese Regeln werden zur Analyse jedes Pakets rekursiv in beide Richtungen durchlaufen. Dabei werden die Chain Options überprüft, welche vom Regel-Parser zur Laufzeit gesetzt wurden. Die erste Regel, welche sich auf ein dekodiertertes Paket anwenden lässt, löst eine entsprechend definierte Aktion aus. [10]

3.5.2 Bro

Bro ist ein Open-Source Network Intrusion Detection System (NIDS), welches durch passives Überwachen des Datenverkehrs Schadsoftware erkennt. Es wurde ausgewählt, da es sich um ein Netzwerk-basiertes Erkennungsprogramm handelt und somit auch die Erkennung von Botnetzen ohne bekannte Signatur möglich ist.

Im Gegensatz zu Snort basiert Bro nicht auf Regeln, sondern benutzt eine Verhaltensüberwachung und Policys zur Botnetz-Erkennung. Dies ermöglicht eine Evaluierung zwischen Bro und anderen Erkennungsprogrammen zur Feststellung, inwiefern sich diese spezifischen Mechanismen auf eine Botnetz-Erkennung auswirken.

Bro kann zur mehrschichtigen Analyse, Verhaltensüberwachung, Policy Enforcement

und Policy-basierten Erkennung von Netzwerkeindringlingen verwendet werden. Außerdem ist mit Bro ein Logging von Netzwerk-Aktivitäten möglich. Dabei werden Eindringlinge entdeckt, indem zunächst der Netzwerkverkehr analysiert wird, um die Semantik auf Anwendungsebene zu gewinnen. Anschließend wird eine Event-orientierte Analyse durchgeführt, welches die Aktivitäten mit dem Muster von Schadsoftware vergleicht. Bei der Analyse von Bro werden sowohl spezifische Angriffe mittels Signaturen-Analyse, als auch Angriffe ohne bekannte Signatur mittels Anomalie-Analyse erkannt.

Bro besteht aus drei größeren Komponenten. Die Datenanalyse erfolgt daher in drei Phasen. Bei der ersten Phase werden pcap Bibliotheken verwendet, um Pakete von den Netzwerk-Interfaces einzufangen. Außerdem wird in dieser Komponente beziehungsweise Phase der Datenverkehr gefiltert und unwichtige Elemente werden verworfen. Der gefilterte Paketstrom wird zur zweiten Komponente weitergeleitet. Diese wird als Bro Event-Engine bezeichnet und interpretiert die Struktur der Pakete, um sie anschließend zu übergeordneten Events zusammenzufassen, welche die durchgeführte Aktion beschreiben. Die Event-Engine benutzt die Programmiersprache *C++*. Zuletzt folgt der Policy Script Interpreter. Dieser verwendet die übergeordneten Events der Event-Engine und vergleicht sie mit den Policy Skripten des Systems. Die Events werden dafür in einer FIFO-Liste sortiert und nacheinander analysiert. Diese Komponente löst verschiedene Aktionen wie Alarmmeldungen aus, wenn ein verdächtiges Paket festgestellt wird und verwirft Events, die nicht in den Policy Scripts definiert sind. Je genauer die Skripts geschrieben sind, desto geringer wird hierbei die Falsch-Positiv-Rate sein.

Bro ist auf Netzwerke mit einer hohen Geschwindigkeit und großen Datenmengen spezialisiert. Daher wird es vor allem von Seiten verwendet, welche ein äußerst flexibles und stark anpassbares Intrusion Detection System benötigen. Bro wurde hauptsächlich für Forschungen bezüglich Eindringungserkennungen und Datenanalysen entwickelt. [11, 26]

3.5.3 Anditoto

Bei Antidoto handelt es sich um ein Open-Source Programm gegen Schadsoftware und Rootkits. Es wurde ausgewählt, da es sich um ein HIDS handelt. Somit kann ein Vergleich der Evaluationskriterien zwischen NIDS und HIDS gewährleistet werden. Außerdem soll es durch Benutzung einer heuristischen Analyse in der Botnetz-Erkennung effektiver als andere Antivirenprogramme sein.

Antidoto entdeckt durch heuristische Analysen des Rechners Viren, Würmer und Botnetze. Dabei überprüft es sowohl verschiedene Dateien auf dem Computer, als auch ungewöhnliche Netzwerkverbindungen. Unter anderem verwendet es dafür eine Whitelist, welche erlaubte Verbindungen und Signaturen enthält. Wird eine Datei oder ein Paket entdeckt, das nicht in dieser Liste enthalten ist, wird vom Programm eine entsprechende Meldung ausgegeben. [27]

3.6 Evaluationskriterien für Erkennungsprogramme

Um Erkennungsprogramme bewerten und untereinander vergleichen zu können, ist die Definition von Evaluationskriterien notwendig. Im Folgenden wurden die Kriterien Effektivität, Effizienz, Ausgabequalität und Konfiguration festgesetzt.

3.6.1 Zuverlässigkeit

Der wichtigste Aspekt eines Erkennungsprogramms ist die Zuverlässigkeit. Ein Programm, welches Botnetze nicht oder nur teilweise erkennt, kann den Rechner oder das Netzwerk nicht zuverlässig vor einer Infizierung schützen. Die Zuverlässigkeit von Erkennungsprogrammen kann durch vier Metriken bestimmt werden, welche alle Möglichkeiten einer Botnetz-Erkennung abdecken.

Falsch-Positiv-Rate: Eine Metrik zur Bestimmung der Zuverlässigkeit eines Erkennungsprogramms ist die Falsch-Positiv-Rate. Hierbei wird eine Botnetz-Infizierung erkannt, obwohl diese nicht vorliegt. Eine hohe Falsch-Positiv-Rate führt zu einer schlechten Zuverlässigkeit, da das Programm nicht zur Unterscheidung zwischen Botnetz- und legitimem Datenverkehr fähig ist.

Falsch-Negativ-Rate: Sofern keine Botnetz-Infizierung erkannt wird, obwohl eine vorliegt, wird dies als Falsch-Negativ-Ergebnis bezeichnet. Diese Metrik stellt die wichtigste Bewertung der Zuverlässigkeit von Erkennungsprogrammen dar, weil Bots im Netzwerk existieren, aber der Nutzer nicht davon benachrichtigt wird. Somit können die Botnetze ungehindert Daten über die Nutzer sammeln und die Rechner für Angriffe verwenden. Eine hohe Falsch-Negativ-Rate resultiert in einer schlechten Zuverlässigkeit.

Wahr-Positiv-Rate: Diese Metrik überprüft, ob eine vorliegende Botnetz-Infizierung vom Erkennungsprogramm erfolgreich erkannt wird. Eine hohe Wahr-Positiv-Rate führt zu einer hohen Zuverlässigkeit.

Wahr-Negativ-Rate: Mit dieser Metrik wird bestimmt, ob das Erkennungsprogramm keine Botnetz-Infizierung erkennt, wenn keine vorliegt. Eine hohe Wahr-Negativ-Rate führt zu einer zunehmenden Zuverlässigkeit.

In Tabelle 3.1 ist eine Übersicht aller vier Metriken dargestellt.

Infizierung	Erkennung	Metrik
✗	✓	Falsch-Positiv
✓	✗	Falsch-Negativ
✓	✓	Wahr-Positiv
✗	✗	Wahr-Negativ

Tabelle 3.1: Metriken zur Bestimmung der Zuverlässigkeit

3.6.2 Effizienz

Die Effizienz ist ein wichtiges Kriterium zur Bewertung von Erkennungsprogrammen, da die verfügbaren Ressourcen auf einem Rechner begrenzt sind. Es wird zwischen Ressourceneffizienz und Erkennungszeiteffizienz unterschieden.

Ressourceneffizienz: Haben die Erkennungsprogramme einen zu hohen Bedarf an Systemressourcen, kann dies den Benutzer in seinen Tätigkeiten einschränken. Auch bei Benutzung eines externen Servers für Erkennungsprogramme ist die die Ressourceneffizienz wichtig, da auch diese in ihrer Leistung begrenzt sind und eine höhere Leistung mit mehr Kosten verbunden ist.

Erkennungszeitseffizienz: Die Erkennungszeiteffizienz bestimmt die Geschwindigkeit, mit der Datenpakete im Netzwerk überprüft werden können. Je schneller die Analyse der Pakete stattfindet, desto mehr Datenverkehr kann überprüft werden, bevor das Erkennungsprogramm Pakete überspringen muss. Außerdem wird hierbei die Dauer bewertet, bis eine Botnetz-Infizierung entdeckt wurde. Je schneller der Nutzer über einen Bot infiziert wird, desto weniger Zeit wird diesem für eine Weiterverbreitung oder das Sammeln von Informationen gegeben.

3.6.3 Ausgabequalität

Ein weiteres Evaluationskriterium ist die Ausgabequalität von Erkennungsprogrammen. Dies ist wichtig, da eine bessere Beschreibung der Meldung mit sehr genauen Informationen über den Befund den Benutzer bei der Identifizierung und Beseitigung der Infizierung unterstützt. Werden zu viele Informationen dargestellt, kann dies zu einer Unübersichtlichkeit führen und eine Beseitigung erschweren. Außerdem ist es wichtig, ob das Erkennungsprogramm gleiche Ereignisse in Gruppen zusammenfassen kann. Da Botnetze eine frequente Kommunikation besitzen können, führt eine Meldung über jedes einzelne Paket ebenfalls zu einer Unübersichtlichkeit. Dies ist besonders in großen Netzwerken ein entscheidender Faktor, wo eine große Anzahl an Rechnern gleichzeitig überwacht wird.

3.6.4 Konfiguration

Da eine Erkennung von Botnetzen in vielen unterschiedlichen Umgebungen erforderlich ist, müssen Erkennungsprogramme durch verschiedene Konfigurationen an die individuellen Gegebenheiten anpassbar sein. Ein Faktor zur Bewertung dieses Kriteriums ist die Anzahl an Möglichkeiten und Übersichtlichkeit der angebotenen Konfigurationen.

Außerdem wird bewertet, wie sich unterschiedliche Konfigurationen auf eine Erkennung auswirken. Dabei wird die Schwierigkeit beurteilt, mit der Nutzer eine wirksame

Konfiguration erreichen können. Dies ist wichtig, da für einen umfassenden Schutz auch verschiedene Nutzer zur Konfiguration eines Erkennungsprogramms fähig sein müssen.

Kapitel 4

Testumgebung

4.1 Testbed

Für die Durchführung der Tests wurde eine gegebene Testumgebung des Lehrstuhls für Netzarchitekturen und Netzdienste an der Technischen Universität München verwendet. Diese kann zur Integration und Simulation von Netzwerk-basierten Anwendungen wie Angriffs- oder Erkennungsprogrammen verwendet werden. Außerdem ist das Erstellen spezifischer Datenströme und Netzwerkstrukturen möglich, um eine realistische Umgebung für die verwendeten Programme zu erzeugen.

Eine Illustration des Testbed-Aufbaus ist in Abbildung 4.1 dargestellt. Die Grundlage bilden zwei Rechner *P1* und *P2*, welche über jeweils zwei Subnetze verfügen. Außerdem sind die jeweiligen Router und das Controlling-Netzwerk dargestellt.

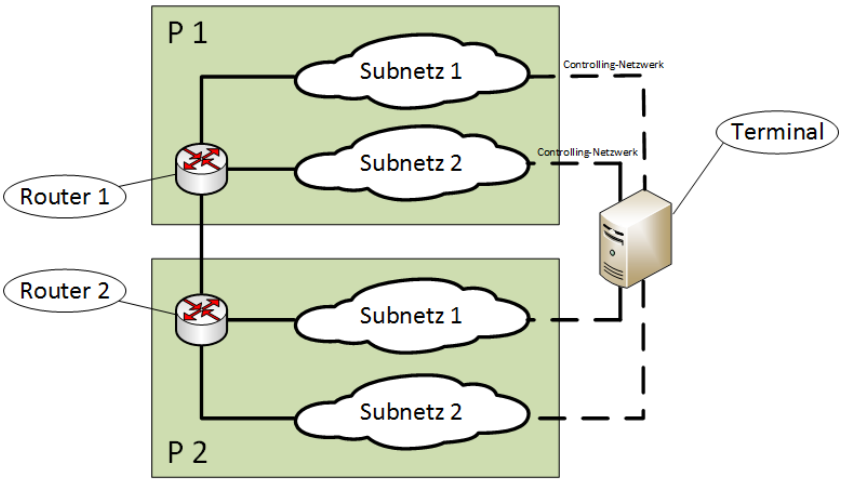

Abbildung 4.1: Testbed des Lehrstuhl

Um in einer Forschungsumgebung eine sorgfältige Analyse des Datenverkehrs ermög-

lichen zu können, ist eine Trennung zwischen durch Testanwendungen erzeugtem Datenverkehr und Netzwerkverkehr zur Kommunikation mit den virtuellen Maschinen erforderlich. Für diesen Zweck existiert in der Testumgebung ein sogenanntes Controlling-Netzwerk. Dieses wird verwendet, um die VMs zu steuern und mit ihnen ohne Beeinträchtigung der Simulationen und Tests kommunizieren zu können.

Von den vier verfügbaren Subnetzen werden nur zwei für die Evaluierung der Botnetz-Software und Erkennungsprogramme verwendet. Die C&C-Server werden in dem einen und die Bots in dem anderen Subnetz platziert. Da Botnetze Weiterverbreitungsroutinen über das Internet besitzen können, wird vor Ausführung der Schadsoftware der Internetzugang blockiert und eine mögliche Verbreitung verhindert.

4.2 Aufbau der Virtuellen Maschinen im Testbed

Zur Gewährleistung einer wissenschaftlichen Testumgebung ist eine Strukturierung der im Testbed verwendeten Rechner notwendig, welche für alle Computer und Programme die gleichen Testbedingungen ermöglicht. Kann dies nicht gewährleistet werden, können die evaluierten Werte verfälscht und nicht mehr miteinander verglichen werden.

Innerhalb des Testbeds handelt es sich bei allen verwendeten Rechnern um virtuelle Maschinen. Um auch hierbei gleiche Testbedingungen zu erzeugen, wurde allen VMs die gleichen Hardware-Ressourcen zugeteilt. Dabei wurde ein Arbeitsspeicher von 2 GB und die Intel(R) Xeon (R) CPU E5-2620 v2 mit 2.10 GHz verwendet.

Nachfolgend ist in Abbildung 4.2 die Benutzung des Testbeds für die Evaluation der verwendeten Programme dargestellt. Für den *Monitoring Rechner* wurde ein Monitoring Port eingerichtet, welcher die Überwachung des gesamten Datenverkehrs in den beiden Subnetzen ermöglicht. Daher werden die Programme Snort und Bro auf diesem Rechner installiert. Die anderen verwendeten Rechner können in zwei Gruppen unterteilt werden. Bei der ersten handelt es sich um Opfer, welche zur Infizierung mit Botnetz-Software benutzt werden. Außerdem dienen diese Rechner dem Testen des Host-basierten Erkennungsprogramms Antidoto. Die zweite Gruppe besteht aus den Command&Control-Servern der beiden Botnetze BeEF und Zeus.

4.2.1 VMs mit Botnetz-Software

Bei den Virtuellen Maschinen der Botnetz-Software wird zwischen C&C-Servern und Opfern unterschieden. Indem sich die Opfer im gleichen Subnetz befinden, wird eine selbstständige Weiterverbreitung auf nicht infizierte Rechner überprüft. Für die beiden Botnetz-Server werden VMs mit Linux Ubuntu 14.10 verwendet. Da diese Maschinen keine grafische Oberfläche besitzen, ist eine Steuerung über das Linux Terminal erforderlich.

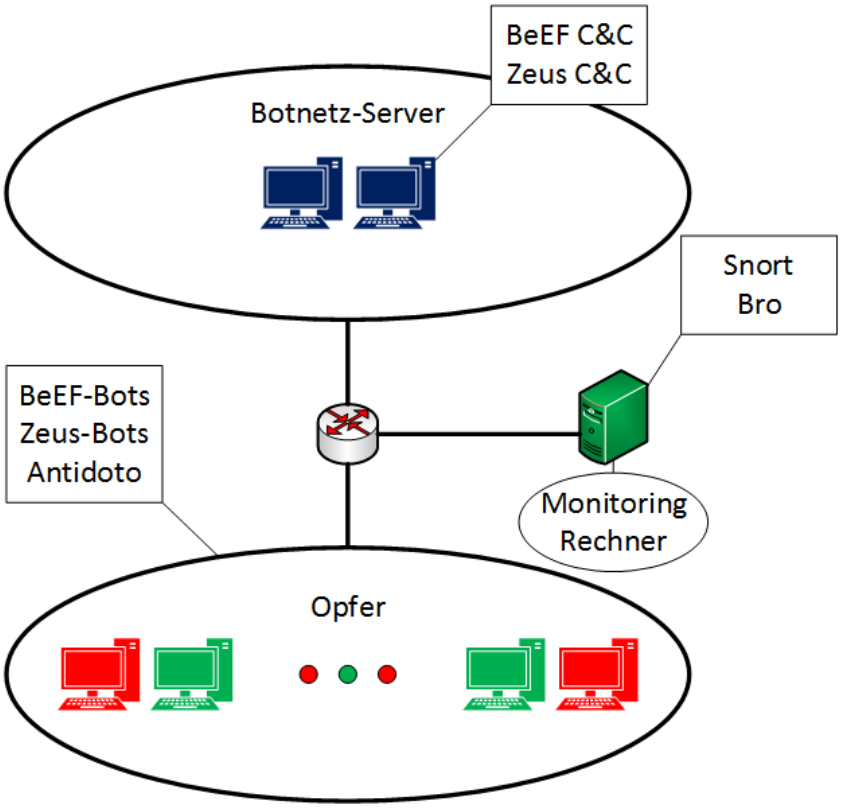

Abbildung 4.2: Aufbau der im Testbed benutzten Rechner. Infizierte Rechner werden rot dargestellt, nicht infizierte grün.

Für die VMs der Botnetz-Opfer wird neben Linux Ubuntu 14.10 auch Windows XP (64-Bit) mit Service Pack 1 und Windows 7 (32-Bit) mit Service Pack 1 verwendet. Grund für die Wahl von Windows ist die nicht mögliche Infizierung von Linux Rechnern mit dem Zeus Botnetz. Außerdem wurden zwei verschiedene Windows-Versionen gewählt, um eine realistische Umwelt und einen Vergleich zwischen beiden Versionen bei der Evaluierung zu ermöglichen.

4.2.2 VMs mit Erkennungssoftware

Neben den Virtuellen Maschinen der Botnetz-Software werden auch VMs mit Botnetz-Erkennungssoftware benötigt. Für die zwei Netzwerk-basierten Programme wird der *Monitoring Rechner* mit dem Betriebssystem Linux Ubuntu 14.10 verwendet. Für das Host-basierte Erkennungsprogramm Antidoto werden infizierte und nicht infizierte Opfer mit Linux Ubuntu 14.10 gewählt. Eine Überprüfung von Windows-Rechnern ist mit diesem Programm nicht möglich.

4.3 Experimente

Um die notwendigen Testdaten für eine Evaluierung der Botnetz-Software und Botnetz-Erkennungsprogramme zu sammeln, wurden einige Experimente durchgeführt, die sich in gezielten Details voneinander unterscheiden. Dies ermöglicht die Überprüfung unterschiedlicher Evaluationskriterien unter verschiedenen Bedingungen. Die durchgeführten Experimente werden entsprechend ihrem Betreff Botnetz-Software oder Erkennungsprogrammen zugewiesen.

4.3.1 Experimente zur Evaluierung von Botnetz-Software

Folgend wird die Feststellung der in Kapitel 3.3 definierten Evaluationskriterien beschrieben.

Funktionsumfang: Zur Überprüfung dieses Kriteriums wird der Umfang angebotener Funktionen und Möglichkeiten einer Erweiterbarkeit bestimmt. Außerdem werden beispielhafte Funktionen der Botnetze getestet und das erzielte Ergebnis bezüglich seiner Korrektheit bewertet. Um eine Unabhängigkeit der Funktionen von spezifischen Betriebssystemen oder Browsern zu gewährleisten, werden mehrere Bots verwendet, welche sich bei diesen Eigenschaften voneinander unterscheiden. Desweiteren ist zum Erreichen realistischer Ergebnisse eine mehrfache Ausführung der Funktionen erforderlich.

Verborgenheit: Zur Feststellung der Verborgenheit werden die Systemressourcen infizierter Rechner auf Unregelmäßigkeiten überprüft. Dabei werden CPU- und Arbeitsspeicherauslastung gemessen, welche durch den Bot entstehen. Außerdem wird das System bezüglich erzeugter Speicherdateien, Prozesse und Threads durch die Infizierung überwacht.

Desweiteren wird die Größe der ausgetauschten Nachrichten und das Updateverhalten der Bots bestimmt. Dabei wird bewertet, ob es sich um ein regelmäßiges Updateintervall handelt und gegebenenfalls die Größe des Intervalls bestimmt.

Weiterverbreitung: Zur Überprüfung dieses Kriteriums wird das Verhalten der Bots auf eigenständige Aktivitäten wie das Scannen ihrer Umgebung überprüft, um eine selbstständige Weiterverbreitung erkennen zu können. Hierfür kann das Netzwerk bezüglich ausgehender Nachrichten von einem Bot überwacht werden. Außerdem ist es möglich, einen Bot mit verschiedenen nicht infizierten Rechnern im gleichen Netzwerk zu testen. Vergrößert sich das Botnetz in diesem Zeitraum, liegt eine selbstständige Weiterverbreitung vor.

Bezüglich einer manuellen Weiterverbreitung werden die Möglichkeiten bewertet, mit denen weitere Rechner infiziert werden können.

Ausfallsicherheit: Zum Bestimmen der Ausfallsicherheit wird die Netzwerk-Struktur des Botnetzes betrachtet, da diese wesentlichen Einfluss auf dieses Kriterium besitzt. Außerdem werden einzelne Bots abgeschalten und die entstehenden Auswirkungen auf das Botnetz überprüft.

Portabilität: Zur Bewertung der Portabilität variieren die infizierten Rechner in ihren Eigenschaften bezüglich des vorliegenden Betriebssystem, ihrer Bit-Architektur und des Browsers.

Anonymität: Um die Anonymität zu bewerten wird die Netzwerk-Struktur des Botnetzes bestimmt und überprüft, ob die Verwendung eines Proxy-Servers unterstützt wird.

Benutzbarkeit: Die Bewertung dieses Kriteriums kann nicht durch ein verändertes Setup beeinflusst werden und ist ausschließlich von subjektiven Eindrücken abhängig.

4.3.2 Experimente zur Evaluierung von Erkennungsprogrammen

Folgend werden Experimente zur Feststellung der in Kapitel 3.6 definierten Evaluationskriterien beschrieben. Anschließend werden in Tabelle 4.1 die durchgeführten Experimente dargestellt.

Zuverlässigkeit: Zur Überprüfung der Zuverlässigkeit wird die Anzahl gleichzeitig verwendeter Botnetze variiert. Dabei muss jedes zu erkennende Botnetz im Einzelnen und in beliebiger Kombination mit anderen Botnetzen getestet werden. Somit kann festgestellt werden, ob das Auftreten mehrerer Botnetze Auswirkungen auf die Zuverlässigkeit der Erkennungsprogramme hat.

Zum Bestimmen der vier Metriken, die in Kapitel 3.6.1 beschrieben worden sind, werden für jedes Botnetz eine unterschiedliche Anzahl an Botnetz-Opfern verwendet. Dabei wird neben der Anzahl der Bots auch die Zahl der nicht infizierten Rechner variiert. Somit kann eine veränderte Zuverlässigkeit festgestellt werden, welche aus einer unterschiedlichen Komposition von Bots und nicht infizierten Rechnern resultiert.

Desweiteren ist die Überprüfung notwendig, inwiefern Hintergrundverkehr im Netzwerk eine Erkennung beeinflusst. Dafür kann Datenverkehr über HTTP, FTP oder VoIP simuliert und alle Experimente zum Testen der Zuverlässigkeit sowohl mit, als auch ohne Hintergrundverkehr durchgeführt werden, um eine mögliche Veränderung festzustellen.

Effizienz: Zur Überprüfung der Ressourceneffizienz wird die CPU- und Arbeitsspeicherauslastung gemessen, welche durch ein Erkennungsprogramm entsteht. Dafür wird der durch das Programm erzeugte Systemprozess bestimmt und die Auslastung gemessen.

Zum Bestimmen der Erkennungszeiteffizienz wird die Anzahl infizierter Bots und ver-

wendeter Botnetze variiert. Dadurch kann überprüft werden, ob eine erhöhte Netzwerkaktivität von Bots die Erkennungszeit beeinflusst.

Ausgabequalität: Zur Überprüfung der Ausgabequalität wird ebenfalls die Anzahl infizierter Bots variiert. Dadurch kann die Übersichtlichkeit und Fähigkeit zur Zusammenfassung gleicher Ereignisse bei einer unterschiedlichen Anzahl an Meldungen bewertet werden.

Konfiguration Zum Bestimmen der Konfigurationen wird betrachtet, wie viele Konfigurationseinstellungen und wie viel Fachwissen über das Netzwerk oder das Erkennungsprogramm für eine zuverlässige Erkennung notwendig sind.

Besitzen die Erkennungsprogramme die Möglichkeit unterschiedlicher Konfigurationen, werden diese bei den Experimenten variiert und eine Beeinflussung der Zuverlässigkeit und Effizienz bewertet.

In Tabelle 4.1 sind neun verschiedene Experimente dargestellt. Dabei variieren die Anzahl der Botnetz-Server, infizierten Bots und Gesamtzahl an Opfern. Zusätzlich ist eine Identifikationsnummer (ID) angegeben, um einen späteren Bezug auf spezifische Experimente zu ermöglichen. Sind bei den Botnetz-Servern die Namen beider Botnetze eingetragen bedeutet dies eine gleichzeitige Benutzung beider Netze.

Mit diesen Experimenten wird eine unterschiedliche Komposition aus Botnetz-Servern und verwendeten Rechnern erreicht. Dies wird zur Überprüfung der Zuverlässigkeit, Effizienz und Ausgabequalität verwendet, indem es die Feststellung ermöglicht, inwiefern sich ein veränderter Netzwerkverkehr der Botnetze auf die Bewertung der einzelnen Kriterien der Erkennungsprogramme auswirkt.

ID	Botnetz-Server	Erkennungsprogramm	Anzahl Bots	Gesamtanzahl Rechner
1	BeEF	Snort, Bro, Antidoto	1	1
2	Zeus	Snort, Bro, Antidoto	1	1
3	BeEF	Snort, Bro, Antidoto	3	3
4	Zeus	Snort, Bro, Antidoto	3	3
5	BeEF	Snort, Bro, Antidoto	3	5
6	Zeus	Snort, Bro, Antidoto	3	5
7	BeEF, Zeus	Snort, Bro, Antidoto	6	6
8	BeEF, Zeus	Snort, Bro, Antidoto	6	10
9	-	Snort, Bro, Antidoto	0	4

Tabelle 4.1: Übersicht über durchgeführte Experimente

Kapitel 5

Evaluierung

Die Evaluierung basiert auf den in Kapitel 3.3 und 3.6 definierten Evaluationskriterien. Eine Bewertung anhand dieser Kriterien wird durch die in Kapitel 4.3 beschriebenen Experimente und die dokumentierten Beobachtungen, welche während der Integration der Programme gemacht werden konnten, ermöglicht. Durch Beobachtungen feststellbare Kriterien werden möglichst objektiv bewertet. Im Folgenden wird zwischen der Evaluation von Botnetz-Software und Erkennungsprogrammen differenziert.

5.1 Evaluierung der Botnetz-Software

Bei der Evaluierung der Botnetz-Software werden die Botnetze bei jedem Kriterium getrennt voneinander bewertet. Abschließend folgt bei den Kriterien eine Zusammenfassung und Bewertung der erfassten Ergebnisse.

5.1.1 Funktionsumfang

Zur Überprüfung dieses Kriteriums wird der Umfang der bereitgestellten Funktionen der Botnetze betrachtet, nachdem diese in das Testbed integriert wurden. Außerdem wird die Erweiterbarkeit und die Zielerreichung bewertet, indem beispielhafte Funktionen praktisch ausgeführt werden.

5.1.1.1 BeEF

Umfang: BeEF besitzt einen Umfang von 241 verschiedenen Modulen, welche mit der Installation des Programmes integriert werden. Diese werden entsprechend ihrer Funktionalität nach den Bereichen Browser, Chrome Extensions, Debug, Exploits, Host, IPEC, Metasploit, Misc, Network, Persistence, Phonegap und Social Engineering sortiert.

Zielerreichung: Zur Überprüfung der Funktionalitäten von BeEF wurden vier Bots verwendet, welche sich in ihrem Betriebssystem und Browser unterscheiden. In Tabelle 5.1 sind diese mit ihren Eigenschaften dargestellt.

	Betriebssystem	Browser
Bot 1	Windows 7	Google Chrome 44
Bot 2	Windows 7	Internet Explorer 8
Bot 3	Windows XP	Google Chrome 44
Bot 4	Linux Ubuntu 14.04	Mozilla Firefox 40

Tabelle 5.1: Betriebssysteme und Browser der Bots

Im Testbed wurde die Funktionalität folgender Module überprüft:

- Erkennen einer Virtualisierung
- Ausführen eines DoS-Angriffs
- Port Scanner

Alle Funktionen wurden dabei zwei Mal für jeden Bot durchgeführt.

In Tabelle 5.2 sind die Ergebnisse der Tests dargestellt. Zu erkennen ist, dass die Erkennung der Virtualisierung bei allen Bots in beiden Tests fehlgeschlagen ist. Dabei ist zu beachten, dass bei allen Bots eine Virtualisierung mit XEN verwendet wurde. Inwiefern andere Virtualisierungen erkannt werden, kann somit nicht beurteilt werden.

Die DoS-Angriffe haben ebenfalls bei allen Bots nicht zum angestrebten Ergebnis geführt. Bei Bot 2 war die Ausführung des Moduls nicht möglich, bei den übrigen Bots war zwar das Modul ausführbar, eine Analyse des Netzwerkverkehrs ergab jedoch keine gesendeten Pakete an das Opfer.

Ein Port Scan war in allen Tests möglich, die Anzahl erkannter Ports variiert jedoch sowohl zwischen den Bots, als auch den Tests. Von den vier geöffneten Ports des gescannten Rechners konnten diese in allen Tests richtig erkannt werden. Allerdings wurden in den Tests noch zusätzliche offene Ports entdeckt, welche nicht korrekt waren.

	Virtualisierung	DoS-Angriff	Erkannte offene Ports
Bot 1	keine Erkennung	Ausführung hat keine Auswirkung	17 (Test 1) 11 (Test 2)
Bot 2	keine Erkennung	Ausführung nicht möglich	39 (Test 1) 47 (Test 2)
Bot 3	keine Erkennung	Ausführung hat keine Auswirkung	5 (Test 1) 26 (Test 2)
Bot 4	keine Erkennung	Ausführung hat keine Auswirkung	23 (Test 1) 7 (Test 2)

Tabelle 5.2: Ergebnisse der getesteten Funktionen von BeEF

Erweiterbarkeit: Da es sich bei BeEF um ein modulares Framework handelt, ist das Hinzufügen eigener Module möglich. Hierfür wird von BeEF eine API zur Entwicklung eigener Funktionen angeboten, wodurch der Entwicklungsaufwand reduziert wird. Um diese nutzen zu können ist eine Authentifikation mittels einem Restful API Key notwendig, welcher beim Start von BeEF zufällig generiert wird. [22]

5.1.1.2 Zeus

Umfang: Da Zeus hauptsächlich zur Spionage sensibler Nutzerdaten verwendet wird, dienen auch die bereitgestellten Funktionen hauptsächlich diesem Zweck. Zeus bietet über den C&C-Server die Möglichkeit an, verschiedene Informationen über seine Nutzer abzufragen. Desweiteren verwendet es einen Keylogger, um die Benutzereingaben der Bots aufzuzeichnen und an den C&C-Server zu übermitteln. In einer Datenbank können daraufhin alle gesammelten Daten abgefragt werden.

Zielerreichung: Alle getesteten Funktionalitäten von Zeus wurden mit zwei infizierten Rechnern durchgeführt, welche das Betriebssystem Windows 7 (32-Bit) mit Service Pack 1 benutzen.

Die abfragbaren Informationen über die Bots betreffen unter anderem das Betriebssystem, das verwendete Service Pack und die verwendete Sprache der Bots und waren bei beiden getesteten Bots korrekt. Desweiteren ist die Darstellung eines Screenshots des Bots möglich. Dabei wird zusätzlich zu den allgemeinen Informationen eines Bots ein Bild über dessen Desktop dargestellt. Dies wird bei jeder Anforderung vom Bot an den C&C-Server übertragen und entspricht immer dem Desktop der Bots zum Zeitpunkt der Anforderung.

Bei der Verwendung des Keyloggers werden die Benutzereingaben eines Bots nach den besuchten Webseiten während der Eingabe sortiert. Ein Eintrag enthält dabei die URL der besuchten Webseite, die zu dieser Zeit getätigten Benutzereingaben und zusätzliche Informationen der Webseite wie die Position von Buttons oder die überprüften Domains. Zum Testen dieser Funktion wurde der Einlogvorgang eines E-Mail Kontos und die Authentifizierung beim Online-Banking einer Bank mit zwei Bots simuliert. Bei beiden konnten die Eingaben erfolgreich aufgezeichnet werden. Beim E-Mail Konto wurden die Daten zusätzlich den Eingabefeldern E-Mail Adresse und Passwort korrekt zugewiesen und entsprachen den tatsächlichen Eingaben. Beim Online Banking war eine korrekte Benennung der Eingabefelder nicht möglich. Eine Trennung zwischen Eingaben für Kontonummer und Eingaben für die PIN war dennoch möglich und richtig. Dies zeigt, dass der Keylogger von Zeus funktioniert und zur Spionage sensibler Daten effektiv verwendet werden kann.

Erweiterbarkeit: Eine Erweiterung der Funktionen von Zeus ist durch das Schreiben eigener Skripte möglich. Hierfür können über den C&C-Server neue Skripte verfasst und

gespeichert werden. Über diese kann das Botnetz zur Ausführung von DDoS-Angriffen befähigt werden.

5.1.1.3 Zusammenfassung und Bewertung

Bei einem Vergleich der beiden Botnetze fällt auf, dass Zeus standardmäßig nur wenig Funktionen bereitstellt, deren Verwendung auf den Informationsdiebstahl spezialisiert ist. BeEF dagegen stellt eine Vielzahl vordefinierter Module zur Verfügung, welche verschiedene Verwendungszwecke abdecken.

Bei Zeus haben die Funktionen ordnungsgemäß funktioniert und in allen Fällen das angestrebte Ergebnis erreicht. Die Module von BeEF haben trotz Bereitstellung verschiedener Browser oftmals ihre Funktion nicht erfüllt.

Die Entscheidung, welches Botnetz einen besseren Funktionsumfang besitzt ist abhängig davon, ob eine größere Anzahl an Funktionen oder eine höhere Effektivität priorisiert wird.

5.1.2 Verborgenheit

Zur Bewertung der Verborgenheit wurde die Persistenz und Unregelmäßigkeiten bei Prozessen oder Systemdaten der Bots überprüft. Außerdem wurde das Updateintervall der Bots mit dem Programm *wireshark* ermittelt, um die Botnetze bezüglich ihrer Netzwerkauffälligkeit zu bewerten.

5.1.2.1 BeEF

Persistenz: BeEF infiziert Rechner durch das Hooken von Webbrowsern. Die Bots können nur gesteuert werden, wenn im Browser die bereitgestellte Webseite des BeEF-Servers angezeigt wird. Wird die Webseite gewechselt, geschlossen oder der Rechner heruntergefahren, ist der Bot nicht mehr steuerbar. Dadurch besitzt BeEF eine schlechte Persistenz und benötigt für eine dauerhafte Kontrolle des Bots eine wiederholte Infizierung. Dies führt zu einer höheren Auffälligkeit für Erkennungsprogramme und wirkt sich negativ auf die Verborgenheit aus.

Hostauffälligkeiten: Durch die Browser-basierte Infizierung findet keine Installation von fremder Software auf infizierten Rechnern statt, welcher zur Aufdeckung der Bots verwendet werden kann. Die Bots werden über Javaskripte kontrolliert, welche vom BeEF-Server heruntergeladen werden. Da einige Browser für jede geöffnete Webseite einen neuen Prozess mit ihrem Namen anlegen, entsteht bei einem BeEF-Bot durch die Tarnung als legitime Webseite kein auffälliger Prozess auf dem Rechner. Die verursachte

Ressourcenauslastung war bei den Browsern Mozilla Firefox, Google Chrome und Internet Explorer schwankend mit maximal 2% CPU-Auslastung und durchschnittlich 0.1% Arbeitsspeicherauslastung. Die verursachte Ressourcenauslastung stellt somit keine Auffälligkeit dar.

Netzwerkauffälligkeiten: Eine Analyse des Update-Intervalls hat ergeben, dass jeder Bot ungefähr jede Sekunde den BeEF-Server kontaktiert, um ein Javaskript anzufordern. Der Server sendet das geforderte Skript daraufhin umgehend zum Bot. Dies stellt eine große Auffälligkeit dar, obwohl Javaskripte auch von legitimen Webseiten im Internet benutzt werden, weil bei mehreren infizierten Rechnern in einem Netzwerk jede Sekunde von allen Bots die gleiche Webseite kontaktiert wird. Dies erzeugt ein kontinuierliches Kommunikationsmuster zwischen den Bots und dem C&C-Server. Dieses regelmäßige Muster kann von Erkennungsprogrammen zur Aufdeckung des Botnetzes genutzt werden.

5.1.2.2 Zeus

Persistenz: Bei der Infizierung eines Rechners mit Zeus wird ein Eintrag in der Windows-Registrierungsdatenbank angelegt, wodurch die Ausführung des Zeus-Bots bei jedem Neustart von Windows ermöglicht wird und der Rechner weiterhin infiziert bleibt. Dies führt zu einer hohen Persistenz von Zeus und erhöht die Verborgenheit, da für eine dauerhafte Kontrolle von Bots keine neue Infizierung notwendig ist.

Hostauffälligkeiten: Folgende Beschreibung bezieht sich auf Bots mit Windows 7 (32-Bit). Zeus besitzt eine ausführbare Datei zur Infizierung neuer Rechner. Wird diese ausgeführt, wird ein Ordner und eine ausführbare Datei im Verzeichnis *C:/Users/Username/AppData/Roaming* angelegt. Bei unterschiedlichen Bots variierte auch die Bezeichnung von Ordner und Datei. Zusätzlich handelt es sich um einen versteckten Ordner, wodurch er nur bei einer erweiterten Ansicht dargestellt wird. In dem Ordner werden die gesammelten Informationen der Bots und die Konfiguration gespeichert. Durch das Anhängen zufällig generierter Bytes an die erzeugten Dateien wird die Prüfsumme dabei künstlich verändert.

Um keinen auffälligen Prozess zu erzeugen, kopiert sich Zeus in den virtuellen Speicher des Prozesses *winlogon.exe* und erzeugt einen neuen Benutzer-Thread. Der Prozess *winlogon.exe* ermöglicht die An- und Abmeldung von Benutzern und läuft durchgehend im Hintergrund mit. Dadurch ist der erzeugte Thread möglichst unauffällig und schwer zu erkennen. Im Testbed konnte nach der Infizierung ein Anstieg des Arbeitsspeicherbedarfs dieses Prozesses um 12 Kilobyte festgestellt werden. Dieser Anstieg ist gering und somit schwer auf einen Bot zurückzuführen. [8] Indem Zeus diese Maßnahmen zur Erhöhung der Verborgenheit verwendet, wird eine Erkennung des Botnetzes erschwert.

Netzwerkauffälligkeiten: Zeus besitzt standardmäßig ein regelmäßiges Update-Intervall

von ungefähr einer Minute. Dies bedeutet, dass jeder Bot des Botnetzes jede Minute den C&C-Server kontaktiert und die binäre Konfigurationsdatei anfordert. Das Update-Intervall kann in der Konfigurationsdatei variiert werden, welche zur Erstellung der ausführbaren Infizierungsdatei der Bots dient. Sind in einem Netzwerk mehrere Zeus-Bots vorhanden, senden diese in gleichen Abständen Pakete an die selbe IP-Adresse. Da dieser Kontakt kontinuierlich stattfindet, kann es vom Erkennungsprogramm als regelmäßiges Muster erkannt werden und steigert die Gefahr einer Aufdeckung.

5.1.2.3 Zusammenfassung und Bewertung

Zeus besitzt eine höhere Persistenz als BeEF, wodurch für eine dauerhafte Kontrolle der Bots keine neue Infizierung notwendig ist und die Verborgenheit erhöht wird. Dafür hinterlässt BeEF keine auffälligen Signaturen auf infizierten Rechnern, wodurch es im Bezug auf die Hostauffälligkeit eine höhere Verborgenheit als Zeus aufweist.

Hinsichtlich des Netzwerkverhaltens besitzen beide Botnetze ein auffälliges Kommunikationsmuster, da sowohl BeEF-, als auch Zeus-Bots in regelmäßigen Abständen Updates durchführen. Dies kann optimiert werden, indem die Bots nur bei Bedarf den C&C-Server kontaktieren. Dadurch würde eine unregelmäßige Kommunikation entstehen und eine Erkennung erschweren.

5.1.3 Weiterverbreitung

Um die Botnetze bezüglich der Weiterverbreitung zu bewerten, wurden sie bezüglich einer selbstständigen und manuellen Weiterverbreitung getestet.

5.1.3.1 BeEF

BeEF besitzt keine selbständige Weiterverbreitungsmöglichkeit. Eine Infizierung neuer Rechner wurde durch das Verschicken der Infizierungswebseiten durchgeführt, welche von BeEF bei einer Installation bereitgestellt werden. Sobald diese in einem Webbrowser geöffnet werden, kann mit dem Rechner über heruntergeladene Javaskripte kommuniziert werden. Eine Infizierung weiterer Rechner ist durch das Verschicken von Links in einer E-Mail möglich. Werden die spezifischen Links von Nutzern angeklickt, öffnet sich im Browser die Webseite und der Rechner ist infiziert.

5.1.3.2 Zeus

Zum Überprüfen der selbstständigen Weiterverbreitungen wurden drei Bots 24 h lang im Testbed mit zwei nicht infizierten Rechnern getestet, von denen einer das Betriebs-

system Windows 7 (32-Bit) und der andere WindowsXP (32-Bit) verwendet. Innerhalb des getesteten Zeitintervalls konnten keine neuen Infizierungen festgestellt werden. Somit konnte für die Zeus-Version 2.0.8.9 keine selbstständige Weiterverbreitung nachgewiesen werden. Eine mögliche Ursache dafür kann der Testaufbau sein, indem sich Zeus zwar nicht zwischen virtuellen Maschinen, aber unter anderen Bedingungen weiterverbreiten kann. Außerdem existieren zahlreiche Versionen von Zeus, wodurch eine selbstständige Weiterverbreitung von anderen Versionen nicht überprüft werden konnte.

Eine eindeutige Überprüfung der selbstständigen Weiterverbreitung von Zeus war mit den eigenen Tests nicht möglich. Wissenschaftliche Quellen, welche eine selbstständige Weiterverbreitung von Zeus belegen oder widerlegen, konnten nicht gefunden werden.

Eine manuelle Weiterverbreitung von Zeus ist über das Schicken von E-Mails möglich, welche in ihrem Anhang die ausführbare Datei zum Infizieren neuer Rechner besitzen.

5.1.3.3 Zusammenfassung und Bewertung

BeEF und Zeus besitzen ähnliche Eigenschaften bezüglich ihrer Weiterverbreitung. Beide konnten sich nicht selbstständig, sondern nur manuell weiterverbreiten. Dadurch sind sie unauffälliger und schwerer zu erkennen als Botnetze, welche aktiv ihre Umgebung scannen und sich selbstständig weiterverbreiten. Dafür ist es mit BeEF und Zeus schwieriger, neue Rechner zu infizieren beziehungsweise mehr Aufwand dafür notwendig.

5.1.4 Ausfallsicherheit

Zur Evaluierung dieses Kriteriums wurde die Netzwerk-Struktur der Botnetze betrachtet, da sich diese wesentlich auf die Ausfallsicherheit auswirkt.

Bei den betrachteten Botnetzen handelt es sich um C&C-Botnetze ohne Bot-Controller. Dadurch kommunizieren die Bots ausschließlich direkt mit dem C&C-Server und nicht untereinander. Wird ein Bot abgeschalten, beeinträchtigt dies nicht die Kommunikation der anderen. Bei C&C-Botnetzen ist somit die vorliegende lokale Transitivität irrelevant zur Bestimmung der Ausfallsicherheit.

5.1.5 Portabilität

Zur Bewertung der Portabilität wurden verschiedene Betriebssysteme und Browser für Zeus beziehungsweise BeEF verwendet.

5.1.5.1 BeEF

Bei BeEF handelt es sich um ein Browser-basiertes Botnetz, wodurch eine Infizierung unabhängig vom Betriebssystem stattfindet. Im Testbed wurden die Webbrowser Mozilla Firefox 40, Google Chrome 44 und Internet Explorer 8 getestet. Bei den getesteten Funktionen konnte kein Unterschied zwischen den verwendeten Browsern festgestellt werden. Für alle Browser war die Ausführung von Kommandos möglich.

5.1.5.2 Zeus

Zeus infiziert Rechner durch eine ausführbare Bot-Datei. Diese ist unter Linux und Mac OS nicht ausführbar. Dadurch ist eine Infizierung dieser Betriebssysteme nicht möglich und auf Windows beschränkt. Eine Infizierung von Rechnern mit Windows 7 war sowohl mit 32-Bit, als auch mit 64-Bit möglich. Dies zeigt, dass die die Infizierung unabhängig von der verwendeten Bit-Architektur von Windows ist. Eine Infizierung eines WindowsXP Rechners mit 32-Bit war im Testbed nicht möglich, da die ausführbare Bot-Datei nicht als valide Anwendung bewertet und somit die Ausführung verhindert wurde. Ein möglicher Grund dafür ist, dass die Erzeugung der Bot-Datei mit einem Windows7-Rechner durchgeführt wurde, wodurch sie keine Kompatibilität mit WindowsXP besitzt. Die Erzeugung der Bot-Datei mit WindowsXP war ebenfalls nicht möglich, da auch dies nicht als valide Anwendung gewertet wurde.

5.1.5.3 Zusammenfassung und Bewertung

BeEF besitzt im Vergleich zu Zeus eine höhere Portabilität als Zeus, da es nicht auf die Infizierung von Windows-Rechnern beschränkt ist. Eine bessere Portabilität ermöglicht die Infizierung einer größeren Anzahl an Rechner und resultiert in einer besseren Weiterverbreitung des Botnetzes.

5.1.6 Anonymität

Zur Evaluierung der Anonymität wurde die Netzwerk-Struktur der Botnetze und die Möglichkeit der Verwendung eines Proxy-Servers betrachtet.

BeEF und Zeus verwenden einen C&C-Server ohne Bot-Controller. Damit gewähren sie eingeschränkte Anonymität für den Botmaster, da die Rückverfolgung eines DDoS-Angriffes mit den Botnetzen nur zur Aufdeckung der beteiligten Bots führt. C&C-Botnetze mit Bot-Controllern und Peer-to-Peer Botnetze besitzen eine höhere Anonymität, da eine Aufdeckung des Botmasters zusätzlich erschwert wird. Somit gewähren BeEF und Zeus im Vergleich mit anderen Netzwerk-Strukturen eine geringe Anonymität für den Botmaster.

Bei BeEF ist die Verwendung eines Proxy-Servers möglich, indem beim C&C-Server ein infizierter Browser als Proxy bestimmt wird. Dadurch kann die Anonymität des Botmasters gesteigert werden.

Zeus unterstützt nicht die Verwendung eines Proxy-Servers durch bereitgestellte Funktionen, was sich negativ auf die Anonymität auswirkt.

5.1.7 Benutzbarkeit

Für die Evaluierung der Benutzbarkeit werden folgende Eigenschaften der Botnetz-Software beschrieben:

- Notwendige Schritte zur Installation
- Benutzerinterface zur Steuerung
- Schwierigkeit der Verwendung
- Dokumentation

5.1.7.1 BeEF

Notwendige Schritte zur Installation: Zur Installation des C&C-Servers wurde im Testbed ein Rechner mit Linux Ubuntu 14.04 gewählt. Die Software kann unter [22] kostenlos heruntergeladen werden. Bevor diese installiert werden kann, ist eine Installation der Programmiersprache Ruby erforderlich. Anschließend ist eine Anpassung der Konfigurationsdatei von BeEF notwendig. In dieser werden unter anderem die IP-Adresse des C&C-Servers, ein Benutzername und Kennwort zur Authentifikation und die verwendeten Module festgelegt. Damit ist die Installation des C&C-Servers abgeschlossen. Die Erzeugung einer Datei zur Infizierung von Rechnern ist bei BeEF nicht notwendig. Dafür benötigte Webseiten werden beispielhaft auf dem C&C-Server angeboten.

Benutzerinterface zur Steuerung: BeEF kann sowohl über ein Terminal, als auch über ein grafisches Benutzerinterface gesteuert werden. Das Benutzerinterface von BeEF ist übersichtlich strukturiert und enthält in einer Liste alle Bots. Diese werden zwischen Online Bots und Offline Bots getrennt mit ihrer IP-Adresse, dem verwendeten Betriebssystem und ihrem Webbrowser dargestellt. Bei Auswahl eines Bots werden erweiterte Informationen bezüglich des Browsers und Betriebssystems dargestellt. Außerdem werden verfügbare Kommandos strukturiert dargestellt und ihre Funktion beschrieben.

Schwierigkeit der Verwendung: Eine Verwendung von BeEF wird dem Benutzer möglichst vereinfacht, indem auf der Startseite des C&C-Servers die Funktionen von BeEF beschrieben und erklärt werden. Außerdem werden Demowebseiten bereitgestellt, welche zur Infizierung von Rechnern verwendet werden können. Bei Bedarf können

hierfür auch eigene Webseiten erstellt werden. Desweiteren besitzt BeEF eine Vielzahl vorinstallierter Module. Um deren Verwendung zu erleichtern, wird ein Ampel-System benutzt, welches jedes Kommando mit einer Farbe markiert. Diese gibt Auskunft darüber, ob das entsprechende Kommando für den Benutzer sichtbar ist, bei dem betreffenden Bot funktioniert oder noch nicht getestet wurde. Soll ein Kommando ausgeführt werden und benötigt zusätzliche Informationen wie die IP-Adresse des Ziels, kann dies über entsprechende Eingabefelder angegeben werden. Nach Abschluss der Durchführung können die Ergebnisse eingesehen und ausgewertet werden. Dies ermöglicht auch die gleichzeitige Durchführung verschiedener Kommandos auf dem selben Bot.

Dokumentation: Für BeEF existiert eine offizielle Vertriebsseite, welche verschiedene Dokumentationen und ein Wiki beinhaltet. In diesen werden die Installation, Funktionen und Module von BeEF ausführlich erläutert. Dies erleichtert die Installation und Handhabung erheblich, da ausführliche Hilfe bei verschiedenen Arten von Problemen übersichtlich angeboten werden.

5.1.7.2 Zeus

Notwendige Schritte zur Installation: Die Installation von Zeus beginnt mit dem C&C-Server. Für diesen wurde im Testbed ein Rechner mit Linux Ubuntu 14.04 gewählt. Zunächst ist ein Webserver notwendig, welcher als C&C-Server fungiert und alle Ressourcen zur Verfügung stellt, die zum Steuern des Botnetzes benötigt werden. Desweiteren ist das Erstellen einer MySQL Datenbank erforderlich, in welche die Informationen der einzelnen Bots eingetragen werden. Beim Herunterladen des Quellcodes für den Zeus-Server ist zu beachten, dass oftmals fehlerhafter Code angeboten wird. Der Quellcode wird anschließend auf den Webserver geladen und für einen Zugriff freigegeben. Folgend wird die Installation des C&C-Servers durch das Aufrufen einer im Quellcode enthaltenen Indexdatei ermöglicht.

Ein Zeus Builder wird benötigt, um die für eine Infizierung benötigte ausführbare Bot-Datei und eine verschlüsselte binäre Konfigurationsdatei zu erstellen. Dafür ist die Angabe einer Konfigurationsdatei notwendig, welche unter anderem Informationen über die Adresse des C&C-Servers und das Update-Intervall der Bots enthält. Die damit erzeugte verschlüsselte Konfigurationsdatei wird auf dem Webserver hinterlegt und im weiteren Verlauf von den Bots angefordert, um Informationen bezüglich ihrer Konfiguration zu erhalten. Die ausführbare Bot-Datei dient der Infizierung neuer Rechner.

Benutzerinterface zur Steuerung: Das Benutzerinterface des C&C-Servers ist übersichtlich dargestellt und enthält verschiedene Informationen des Botnetzes wie die Anzahl aktiver Bots. Außerdem wird eine Ansicht für nähere Details der Bots und ein Menü zum Durchsuchen der MySQL Datenbank angeboten. In der Datenbank kann nach spezifischen Bots oder bestimmten Zeiträumen gefiltert werden. Die Sammlung

der Benutzereingaben erfolgt über den integrierten Keylogger der Bots und wird in der Datenbank nach den besuchten Webseiten und dem jeweiligen Datum strukturiert.

Schwierigkeit der Verwendung: Funktionen des Zeus Botnetzes wie das Aufzeichnen von Benutzereingaben sind leicht bedienbar, da die Eingaben automatisch von den Bots gesammelt und an den C&C-Server geschickt werden. Dort können sie vom Botmaster eingesehen und für einen Missbrauch verwendet werden. Die Verwendung eigener Funktionen wird unterstützt, indem über den Server direkt neue Skripte definiert, gespeichert und spezifischen Bots zugewiesen werden können.

Dokumentation: Über die Installation und Verwendung des Zeus Botnetzes sind zahlreiche Dokumentationen im Internet erhältlich. Dabei werden diverse Tutorials über die Installation von Zeus und Beispiele für diverse Verwendungsmöglichkeiten angegeben. Für Zeus ist keine offizielle Dokumentation erhältlich. Dies führt dazu, dass vorhandene Dokumente oft Fehler enthalten und teilweise wichtige Details bei der Installation auslassen, wodurch die Installation und Steuerung von Zeus erschwert wird. Außerdem wird in den meisten Anleitungen Windows als Betriebssystem für den C&C-Server verwendet. Da im Testbed hierfür Linux Ubuntu verwendet wurde und damit der Installationsprozess stark von den Dokumentationen abwich, ist eine hohe Einarbeitungszeit zum Etablieren des Zeus Botnetzes notwendig

5.1.7.3 Zusammenfassung und Bewertung

Im Gegensatz zu BeEF ist die Installation von Zeus erheblich zeitaufwändiger. Dies liegt zum einen an den zusätzlichen benötigten Programmen für den Zeus C&C-Server und zum anderen an dem nicht offiziellen Vertrieb des Quellcodes. Dieser war im Testbed oftmals fehlerhaft, wodurch eine Installation erschwert wurde. Außerdem existiert eine Vielzahl fehlerhafter Anleitungen zur Installation im Internet, was eine Integration in die Testumgebung zusätzlich erschwerte. BeEF dagegen besitzt eine offizielle Webseite für das Herunterladen des Programms und ausführliche Installationsanleitungen, sodass eine Verwendung des Programms erleichtert wurde.

Beide Botnetze besitzen einen C&C-Server mit einem übersichtlichen grafischen Benutzerinterface zur Steuerung der Bots, wodurch die Kontrolle auch über eine große Anzahl an Bots möglich ist. Die Verwendung der Funktionen stellte bei beiden Botnetzen keine Schwierigkeit dar.

5.2 Evaluation der Erkennungssoftware

Folgend werden die verwendeten Erkennungsprogramme nach den verschiedenen Evaluationskriterien bewertet. Anschließend folgt eine Zusammenfassung und Bewertung

der erzielten Ergebnisse. Bei den Erkennungsprogrammen wurde Snort Version 2.9.6.0, Bro Version 2.4.1 verwendet. Für Antidoto existieren keine unterschiedlichen Versionen.

5.2.1 Zuverlässigkeit

Zur Überprüfung der Zuverlässigkeit wurden die in Tabelle 4.1 aus Kapitel 4.3.2 dargestellten Experimente durchgeführt. Außerdem wurde ein Webserver verwendet, um eine Beeinflussung der Ergebnisse durch HTTP- und HTTPS-Datenverkehr im Netzwerk zu überprüfen.

5.2.1.1 Snort

Snort ermöglicht eine Erkennung von Schadsoftware durch das Verwenden von Regeln. Daher wurde eine Vielzahl unterschiedlicher Regelsätze in Snort integriert, um eine möglichst hohe Zuverlässigkeit zu erreichen. Dabei wurden sowohl allgemeine Regeln der Snort Community, als auch spezifische auf Schadsoftware und Zeus spezialisierte Regeln verwendet.

Eine Analyse der Regeln, welche speziell zur Erkennung von Botnetzen verwendet werden, ergab eine Fokussierung auf die Command & Control-Server von C&C-basierten Botnetzen. Es existieren veröffentlichte Listen von bekannten aktiven Botnetz-Servern. Diese Listen werden für die Botnetz-Regeln von Snort verwendet, indem die Zieladresse ausgehender TCP- oder UDP-Pakete des Netzwerks hinsichtlich spezifischer IP-Adressen und Ports überprüft wird. Besitzt ein Paket eine solche Adresse mit entsprechender Port-Nummer, wird es von Snort verworfen oder eine Alarmmeldung an den Benutzer ausgegeben.

Regeln, welche speziell zur Erkennung von Zeus angeboten werden, überprüfen ebenfalls die Zieladresse ausgehender Netzwerkpakete bezüglich bekannter Adressen von Zeus C&C-Servern. Zusätzlich wird der Inhalt der Pakete hinsichtlich verschiedener Byte-Folgen und auftretender bekannter DNS-Namen untersucht.

Bei der Durchführung der festgesetzten Experimente war eine Erkennung der Botnetze BeEF oder Zeus mit den im Internet erhältlichen Regeln nicht möglich. Dabei war es irrelevant, ob beide Botnetze einzeln verwendet oder die Komposition von Bots und der Gesamtanzahl verwendeter Rechner variiert wurde. Alle Experimente wurden außerdem mit unterschiedlichen Überwachungszeiten von 1h, 8h und 24h durchgeführt, um die Erkennung eines regelmäßigen Kommunikationsmusters der Bots zu ermöglichen.

Eine Erkennung der Botnetze war mit Snort erst nach einer Anpassung der Regeln möglich, indem die IP-Adressen und Ports der Botnetz-Server zum Regelsatz hinzugefügt oder die Überprüfung der Paketinhalte gezielt auf die ausgetauschten Pakete der Bots festgelegt wurde. Mit diesen Veränderungen konnte Snort die Botnetze zwar zuverlässig

erkennen, jegliche Veränderung der IP-Adressen, Ports oder Paketinhalte erfordert allerdings eine erneute Anpassung der Regeln.

5.2.1.2 Bro

Zur Bewertung der Zuverlässigkeit von Bro wurden die standardmäßig integrierten Policys zur Verhaltensüberwachung des Netzwerks überprüft. Diese wurden dazu konzipiert, die meisten Internet-Angriffe zu erkennen und ermöglichen unter anderem eine Erkennung von Schadsoftware anhand bekannter Hash-Werte oder Hostnamen. [28]

Die Durchführung der definierten Experimente mittels der integrierten Policys führte zu keiner Erkennung der Botnetze BeEF und Zeus. Zwar konnte der Datenverkehr bezüglich ein- und ausgehender HTTP-Verbindungen gefiltert werden, jedoch wurde keine Meldung über erkannten Botnetz-Verkehr ausgegeben. Auch eine Erhöhung der Bot-Anzahl für jedes Botnetz zur Erzeugung eines auffälligeren Kommunikationsmusters verursachte keine entsprechende Meldung von Bro.

5.2.1.3 Antidoto

Da es sich bei Antidoto um ein HIDS handelt, welches ausschließlich für Linux-Rechner angeboten wird, konnte es nur bezüglich einer Erkennung von BeEF getestet werden. Dafür wurden Rechner mit dem Betriebssystem Ubuntu 14.04 überprüft. Die Ergebnisse der durchgeführten Experimente zur Überprüfung der Zuverlässigkeit von Antidoto sind in Tabelle 5.3 dargestellt. Unterschieden wurde dabei zwischen einem Rechner, auf dem der BeEF C&C-Server betrieben wurde, einem BeEF-Bot und einem nicht infizierten Rechner. Die Ergebnisse zeigen, dass Antidoto auf allen drei Rechnern jeweils zwei Mal getestet wurde und in keinem Fall eine Meldung über ein erkanntes Botnetz erzeugte.

	Test 1	Test 2
C&C-Server	✗	✗
Bot	✗	✗
Nicht infizierter Rechner	✗	✗

Tabelle 5.3: Evaluierungsergebnisse der Zuverlässigkeit von Antidoto

5.2.1.4 Zusammenfassung und Bewertung

Bei einem Vergleich der Evaluierungsergebnisse fällt auf, dass alle drei Erkennungsprogramme ohne spezifische Anpassungen nicht zur Erkennung von Botnetzen fähig waren und somit keinen zuverlässigen Schutz bieten.

Bei Snort konnte eine Erkennung gewährleistet werden, nachdem die Signatur-basierten Regeln auf die spezifischen Pakete festgelegt wurden. Ohne diese Anpassung waren die Regeln zu allgemein und führten bei den getesteten Botnetzen zu keiner Erkennung. Da die in der Testumgebung verwendeten Botnetze über keine bekannte IP-Adresse verfügen, waren die meisten Regeln wirkungslos. Zusätzlich verwendet das Zeus Botnetz unterschiedliche Ports für die Kommunikation, wodurch Snort-Regeln für spezifische Ports ebenfalls nicht wirken. Dies zeigt, dass unbekannte Botnetze und veränderte Signaturen von Snort nicht zuverlässig erkannt werden können und die meisten Regeln durch veränderte IP-Adressen, Ports oder Hostnamen der Botnetze wirkungslos werden.

5.2.2 Effizienz

Zur Bewertung der Effizienz dienen die in Tabelle 4.1 aus Kapitel 4.3.2 dargestellten Experimente. Mit den Programmen Snort und Bro wurde der selbe Netzwerkverkehr analysiert, um eine Unabhängigkeit der Ergebnisse vom Datenaufkommen zu gewährleisten.

5.2.2.1 Snort

Die durch Snort entstandene CPU-Auslastung war schwankend mit einem Maximum von 0.7 %. Die Arbeitsspeicherauslastung betrug in einem Zeitraum von 8h durchschnittlich 4.4%. Die Ressourcenauslastung wurde dabei mit den Experimenten bestimmt, welche die IDS 3,4 und 7 besitzen und eine unterschiedliche Bot-Anzahl verwenden. Der dadurch verschieden große Netzwerkverkehr führte zu keiner Beeinflussung des Ressourcenbedarfs von Snort.

Indem Snort den Netzwerkverkehr in Echtzeit überwacht, wird eine entsprechende Meldung über eine Botnetzerkennung ausgegeben, sobald eine Regel mit den Informationen oder dem Inhalt eines Netzwerkpakets übereinstimmt. Auch bei einer größeren Anzahl verwendeter Bots, konnten die entsprechenden Meldungen sofort erzeugt werden.

5.2.2.2 Bro

Die Ergebnisse der Ressourceneffizienz von Bro sind in Tabelle 5.4 dargestellt. Für die Bro-Versionen 2.0, 2.1 und 2.4.1 wurde die durchschnittlich erzeugte Auslastung von CPU und Arbeitsspeicher in einem Zeitraum von 8h bestimmt. Um einen Vergleich dieser Werte mit anderen Erkennungsprogrammen ermöglichen zu können, wurde zusätzlich der Bedarfsdurchschnitt von allen drei Versionen bestimmt. Eine Veränderung der Bot-Anzahl führte zu keiner auffälligen Änderung des Ressourcenbedarfs von Bro.

	CPU-Auslastung	Arbeitsspeicher-Auslastung
Bro 2.0	15.8 %	1.0 %
Bro 2.1	16.1 %	0.6 %
Bro 2.4.1	16.6 %	2.1 %
Durchschnitt gesamt	16.36 %	1.23 %

Tabelle 5.4: Ressourcenauslastung von Bro

Bro kann bezüglich seiner Erkennungszeiteffizienz nicht beurteilt werden, da im Testbed keine Erkennung von Botnetz-Datenverkehr möglich war.

5.2.2.3 Antidoto

Antidoto kann bezüglich seiner Ressourceneffizienz nicht bewertet werden, da es als HIDS nur kurzfristig den Rechner und das Netzwerk nach Signaturen durchsucht und somit keine Systemressourcen dauerhaft beansprucht. Die Erkennungseffizienz kann ebenfalls nicht bestimmt werden, da eine Erkennung von Botnetzen in allen Experimenten nicht möglich war.

5.2.2.4 Zusammenfassung und Bewertung

Bei einem Vergleich von Bro und Snort erzeugen beide Programme eine sehr niedrige Arbeitsspeicherauslastung. Bezüglich der CPU wird durch die Verwendung von Snort ebenfalls eine sehr geringe Auslastung erzeugt. Bro dagegen besitzt einen erheblichen CPU-Bedarf, welcher um ein Vielfaches größer als von Snort ist. Somit führt die Benutzung von Snort nur zu geringen Einschränkungen für den Benutzer eines Rechners. Die Verwendung von Bro kann aufgrund der CPU-Auslastung das Nutzerverhalten einschränken. Desweiteren waren bei beiden Programmen keine auffälligen Veränderungen des Ressourcenbedarfs feststellbar, wenn die Bot-Anzahl und damit der zu überprüfende Netzwerkverkehr variiert wurde. Somit kann sowohl Snort, als auch Bro zur Überwachung von auch größeren Netzwerken verwendet werden.

5.2.3 Ausgabequalität

Um die Ausgabequalität der Erkennungsprogramme zu bewerten, wurden verschiedene Meldungen mit den Programmen erzeugt. Außerdem wurde die Experimente der Tabelle 4.1 mit den IDs 3,4 und 7 durchgeführt, um die Übersichtlichkeit der Ausgaben zu beurteilen.

5.2.3.1 Snort

Snort besitzt verschiedene Möglichkeiten, um den Benutzer über erkannte Schadsoftware zu informieren. Dabei können Meldungen entweder an das Syslog gesendet, über die Konsole des Rechners ausgegeben oder in einer separaten Datei gespeichert werden. Bei Zweiterem kann zwischen *Fast alerts* und *Full alerts* unterschieden werden.

Ein Beispiel für den Vergleich dieser beiden Ausgabemöglichkeiten ist in Tabelle 5.5 dargestellt. *Fast alerts* enthalten dabei Basisinformationen über die Regel, welche die Meldung ausgelöst hat und über das betreffende Netzwerkpaket. Diese enthalten unter anderem den Zeitpunkt der Erkennung, die Snort-Regel-ID und die Quell- beziehungsweise Ziel-Adresse des auslösenden Pakets. *Full alerts* enthalten weitere Informationen über das betroffene Netzwerkpaket wie dessen TTL-Nummer, Flags oder die MSS.

Fast alert	09/22-11:47:07.845266 [**] [1:123456:0] Zeus-Bot [**] [Priority: 0] {TCP} 172.16.151.104:80 ->172.16.152.114:51015
Full alert	[**] [1:123456:0] Zeus-Bot [**] [Priority: 0]09/22-11:49:07.870104 172.16.151.104:80 ->172.16.152.114:51017 TCP TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF ***A**S* Seq: 0xE77A8633,Ack: 0x71F346A2,Win: 0x7210,TcpLen: 32 TCP Options (6) =>MSS: 1460 NOP NOP SackOK NOP WS: 7

Tabelle 5.5: Beispiel für eine Konsolenausgabe von Snort mit Fast alerts und Full alerts

Indem Snort die Meldungen durch Absätze voneinander trennt und zeitlich sortiert in einer Datei abspeichert, wird eine übersichtliche Auskunft über erkannte Schadsoftware ermöglicht. Durch das Anbieten verschiedener Ausgabemöglichkeiten kann Snort sowohl für Netzwerke mit einem kleinen Datenverkehr, als auch für sehr große Netzwerke verwendet werden. Falls nur wenig Meldungen analysiert werden sollen oder erweiterte Informationen der Pakete benötigt werden, ist die Verwendung von *Full alerts* von Vorteil. Bei einer Vielzahl an Meldungen oder Netzwerken mit einer großen Anzahl an Rechnern ist dagegen die Verwendung von *Fast alerts* vorteilhaft, da sie eine bessere Übersicht gewähren. Eine Ausgabe über die Konsole enthält die gleichen Informationen der Regeln und Pakete wie *Fast alerts*.

5.2.3.2 Bro

Wird ein Netzwerk mit Bro überwacht, erstellt dieses diverse LOG-Dateien, in denen die Netzwerk-Aktivitäten protokolliert werden. Unter anderem wird in diesen der gesamte HTTP oder DNS-Datenverkehr getrennt voneinander aufgezeichnet. Bei Erkennung eines potenziell interessanten Pakets wird dieses entsprechend gekennzeichnet und kann abhängig von der verwendeten Konfiguration als Gefährdung eingestuft werden. Dabei kann die entsprechende Netzwerkaktivität entweder in einer separaten Log-Datei

gespeichert oder an eine festgelegte E-Mail Adresse geschickt und auf der Konsole des Rechners ausgegeben werden. [26]

Eine solche Log-Datei enthält dabei vielfältige Informationen über die auslösende Netzwerkaktivität wie den Zeitpunkt der Erkennung, die Quell- beziehungsweise Ziel-Adresse des Pakets, die Ports und eine Problembeschreibung, welche die Erkennung ausgelöst hat. Alle Pakete werden getrennt voneinander dargestellt und die zugehörigen Informationen durch Absätze separiert, sodass eine übersichtliche Darstellung über die geloggten Pakete gewährleistet wird. Dadurch ist es möglich, eine große Anzahl an Paketen im Netzwerk aufzuzeichnen und dennoch eine übersichtliche Analyse von aufgetretenen Gefährdungen zu ermöglichen.

5.2.3.3 Antidoto

Wird Antidoto ausgeführt, durchsucht es den Rechner nach spezifischen Signaturen von Schadsoftware. Nachfolgend ist ein Beispiel für eine Ausgabe von Antidoto dargestellt.

> We got warning about process: 'we found SUID (0) or SGID bit (1) enabled, it's very dangerous'
> pid: 2780 name: gnome-pty-helpe ppid: 2774 uid: 1000 gid: -1 exe path: /usr/lib/libvte-2.90-9/gnome-pty-helper

Dabei ist zu erkennen, dass die Gefährdungsstufe und zusätzliche Informationen über den Befund dargestellt werden. Desweiteren wird der Dateipfad angegeben, um eine Lokalisierung und Analyse des Problems zu ermöglichen.

5.2.3.4 Zusammenfassung und Bewertung

Snort und Bro besitzen beide eine übersichtliche Darstellung über potenzielle Schadsoftware, sodass auch große Datenmengen analysiert werden können. Außerdem beinhalten die Ausgaben beider Programme detaillierte Informationen der Pakete, sodass eine Zuordnung zu den Verursachern dieser Pakete ermöglicht wird.

Mit Antidoto ist nur die Überwachung von jeweils einem Rechner möglich. Dadurch ist es für große Netzwerke trotz seiner übersichtlichen und detaillierten Ausgabe möglicher Gefährdungen ungeeignet.

5.2.4 Konfiguration

Zur Bewertung dieses Kriteriums wird die Anzahl verfügbarer Konfigurationen bestimmt, mit denen die Programme an ihre Umgebung angepasst werden können. Außerdem wird festgestellt, wie viele Konfigurationen für eine zuverlässige Verwendung der Erkennungsprogramme notwendig sind.

5.2.4.1 Snort

Snort besitzt eine einzige Konfigurationsdatei, in der alle Einstellungen des Programms festgelegt werden. In dieser können die Netzwerk-Variablen, der Decoder, die Basis Detection Engine, Bibliotheken, Präprozessoren, Ausgabemodule und Regeln von Snort konfiguriert werden. Standardmäßig ist eine Vielzahl der Module deaktiviert. Diese sind zwar in der Konfigurationsdatei enthalten, aber auskommentiert. Sollen sie verwendet werden, ist ein Entfernen der Kommentarzeichen notwendig.

Notwendig für eine zuverlässige Erkennung von Snort ist die Festlegung der Netzwerkadressen der privaten Rechner, sodass zwischen eingehenden und ausgehenden Netzwerkverbindungen unterschieden werden kann. Desweiteren ist die Angabe der Dateipfade der Snort Regeln notwendig, welche für die Detection Engine oder die Präprozessoren verwendet werden sollen.

Die Angabe von Regeln, welche von Snort zur Erkennung von Schadsoftware verwendet werden, ist essenziell, da diese in der Installation von Snort nicht enthalten sind. Die Regeln können entweder unter Verwendung der von Snort definierten Syntax selbst definiert oder im Internet heruntergeladen werden. Da Snort auf diesen Regeln basiert, ist eine Erkennung von Schadsoftware ohne sie nicht möglich. Je genauer die Regeln definiert sind, desto zuverlässiger ist die Erkennung von Snort. [28]

5.2.4.2 Bro

Bei Bro werden drei Konfigurationsdateien verwendet, in denen die Einstellungen bestimmt werden. In diesen werden unter anderem die Netzwerkeinstellungen, Mail Optionen und Logging Optionen festgelegt. Die einzelnen Einstellungsmöglichkeiten besitzen dabei Kommentare über ihre Funktion und sind auskommentiert oder mit Standardwerten versehen, sodass eine Benutzung und Anpassung von Bro an individuelle Bedingungen erleichtert wird.

Für eine Verwendung von Bro ist das Festlegen des zu überwachenden Interfaces und des Hostrechners erforderlich. Außerdem müssen die IP-Adressen der privaten Rechner definiert werden, um zwischen eingehendem und ausgehendem Datenverkehr differenzieren zu können. Die Angabe einer E-Mail Adresse ist zum Senden von Berichten über

auffällige Netzwerk-Pakete notwendig.

Bro besitzt mit der Installation eine Reihe diverser Policy Skripte, welche in der eigenen Sprache von Bro geschrieben sind und zur Erkennung verschiedener Internetangriffe verwendet werden können. Desweiteren sind im Internet Programme erhältlich, welche mit Bro verwendet werden können, um eine umfassendere Erkennung zu ermöglichen. Ein Beispiel dafür ist Botflex, welches die Verhaltensanalyse von Bro mit verschiedenen Komponenten wie Blacklists oder eingehenden Scans in Zusammenhang setzt, um Schadsoftware und speziell Botnetze im Netzwerk zu erkennen. Um Auswirkungen dieses Tools auf die Evaluationsergebnisse von Bro zu überprüfen, wurden Botflex und Botflex-with-correlation-framework mit den Bro-Versionen 2.0, 2.1 und 2.4.1 getestet. Das Auftreten zahlreicher Fehler im Quellcode beider Botflex-Versionen unabhängig von der verwendeten Bro-Version verhinderte eine Evaluierung von Bro mit diesem Tool. Daher ist keine Bewertung der Evaluierungskriterien von Bro mit dem auf Botnetze spezialisierten Botflex möglich. [29]

5.2.4.3 Antidoto

Antidoto besitzt keine Konfigurationsmöglichkeiten, um das Programm bezüglich der individuellen Umgebung anzupassen. Für eine Verwendung des Programms sind daher keine vorherigen Konfigurationen notwendig.

5.2.4.4 Zusammenfassung und Bewertung

Bei einem Vergleich von Snort und Bro bezüglich ihrer Konfigurationen ist erkennbar, dass beide Programme vielfältige Möglichkeiten zur Anpassung an individuelle Bedingungen bereitstellen. Außerdem besitzen sie Erklärungen für die einzelnen Einstellungen und ermöglichen durch Standardwerte eine Verwendung ohne die Notwendigkeit vorheriger umfangreicher Konfigurationen.

Kapitel 6

Fazit

Für die Evaluierung von Botnetzen und Erkennungsprogrammen ist die Definition von Evaluationskriterien notwendig, um die Programme bezüglich verschiedener Aspekte bewerten zu können. Aufgrund einer großen Vielzahl an sowohl Botnetzen, als auch Erkennungsprogrammen wird die Festlegung von Kriterien, welche auf alle Programme anwendbar sind, erschwert.

Bei der Integration der Botnetze in die Testumgebung wies die Installation von Zeus im Vergleich zu BeEF einen erheblich höheren Zeitbedarf auf. Grund dafür ist die schlechte Dokumentation des Programms und der oftmals fehlerhafte Quellcode, der im Internet verfügbar ist.

Auch bei den Erkennungsprogrammen war ein unterschiedlicher Zeitbedarf zur Integration der Programme erkennbar. Während Antidoto keine Installation oder Konfiguration benötigt, sind bei den Netzwerk-basierten Programmen Snort und Bro diverse Konfigurationen und Anpassungen an die Umgebung notwendig, um eine Erkennung zu gewährleisten.

Basierend auf den Evaluierungsergebnissen besitzen BeEF und Zeus sowohl Vor- als auch Nachteile. Während BeEF zur Infizierung verschiedener Betriebssysteme verwendet werden kann, ist mit Zeus nur eine Infektion von Windows-Rechnern möglich. Dafür kann Zeus im Vergleich zu BeEF eine höhere Effektivität vorweisen, da integrierte Funktionen besser das angestrebte Ergebnis erreichen. Die Wahl des besseren Botnetzes ist daher von den Anforderungen und Verwendungszwecken abhängig, die es erfüllen soll.

Bei den Evaluationsergebnissen der Erkennungsprogramme ist erkennbar, dass den drei Programmen Bro, Snort und Antidoto unabhängig von den verwendeten Verfahren oder der Platzierung eine Erkennung der Botnetze nicht möglich war. Größter Vorteil von Snort und Bro ist im Gegensatz zu Antidoto die mögliche Überwachung von großen Netzwerken, da diese auf nur einem zentralen Rechner installiert werden.

Die Evaluierung dieser Arbeit könnte durch erweiterte Konfigurationen der Erken-

nungsprogramme fortgeführt werden. Dies kann in einer höheren Zuverlässigkeit der Programme resultieren und eine Erkennung von Botnetzen mittels regelmäßiger Kommunikationsmuster ermöglichen. Außerdem könnten noch weitere Botnetze und Erkennungsprogramme in eine Testumgebung integriert und evaluiert werden, um eine erhöhte Allgemeingültigkeit der Ergebnisse zu gewährleisten.

Literaturverzeichnis

[1] P. Gibbs, “Botnet Tracking Tools,” August 2014.

[2] M. S. Tjoa and W. Reidlinger, “Aktuelle Verfahren zur P2P Botnetzerkennung und–Bekämpfung.”

[3] S. M. Specht and R. B. Lee, “Distributed denial of service: Taxonomies of attacks, tools, and countermeasures.” in *ISCA PDCS*, 2004, pp. 543–550.

[4] S. Ben, “P2p-botnetz-analyse-waledac.” Universität Mannheim Lehrstuhl Praktische Informatik 1, 2009, p. 52.

[5] ZeitOnline. "CyberBerkut bekennt sich zu Angriff auf Internetseite des Bundestags". Abgerufen am 22.09.2015. [Online]. Available: http://www.zeit.de/digital/internet/2015-01/bundestag-bundeskanzlerin-cyberberkut-angriff-webseiten

[6] D. Dagon, G. Gu, C. P. Lee, and W. Lee, “A taxonomy of botnet structures,” in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual.* IEEE, 2007, pp. 325–339.

[7] B. Lundeen and J. Alves-Foss, “Practical clickjacking with beef,” in *Homeland Security (HST), 2012 IEEE Conference on Technologies for.* IEEE, 2012, pp. 614–619.

[8] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, “On the analysis of the zeus botnet crimeware toolkit,” in *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on.* IEEE, 2010, pp. 31–38.

[9] H. Zeidanloo, M. Shooshtari, P. Amoli, M. Safari, and M. Zamani, “A taxonomy of Botnet detection techniques,” in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 2, July 2010, pp. 158–162.

[10] M. Roesch *et al.*, “Snort: Lightweight intrusion detection for networks.” in *LISA*, vol. 99, no. 1, 1999, pp. 229–238.

[11] P. Mehra, “A brief study and comparison of snort and bro open source network intrusion detection systems,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, no. 6, pp. 383–386, 2012.

[12] T.-F. Yen, "Detecting stealthy malware using behavioral features in network traffic," 2011.

[13] J. Nazario, "Bot and botnet taxonomy," *Computer Security Institute. Computer Security Institute Security Exchange*, 2008.

[14] P. Wang, L. Wu, B. Aslam, and C. C. Zou, "A systematic study on peer-to-peer botnets," in *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on.* IEEE, 2009, pp. 1–8.

[15] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang, "Botnet: Classification, attacks, detection, tracing, and preventive measures," in *Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control*, ser. ICICIC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1184–1187. [Online]. Available: http://dx.doi.org/10.1109/ICICIC.2009.127

[16] R. A. Rodríguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro, "Survey and taxonomy of botnet research through life-cycle," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 45, 2013.

[17] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 2, pp. 113–127, 2010.

[18] M. Eslahi, R. Salleh, and N. B. Anuar, "Bots and botnets: An overview of characteristics, detection and challenges," in *Control System, Computing and Engineering (ICCSCE), 2012 IEEE International Conference on.* IEEE, 2012, pp. 349–354.

[19] D. Dittrich and S. Dietrich, "P2p as botnet command and control: a deeper insight," in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on.* IEEE, 2008, pp. 41–48.

[20] W. C. Hanna, "Using snort to detect rogue irc bot programs," 2004.

[21] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 41–52. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177086

[22] The Browser Exploitation Framework Project. Abgerufen am 25.08.15. [Online]. Available: https://github.com/beefproject/beef/wiki

[23] ZeuS Tracker. Abgerufen am 20.08.15. [Online]. Available: https://zeustracker.abuse.ch/blocklist.php

[24] C. H. Rowland, "Intrusion detection system," Jun. 11 2002, uS Patent 6,405,318.

[25] Snort. Abgerufen am 27.08.15. [Online]. Available: https://www.snort.org

[26] The Bro Network Security Monitor. Abgerufen am 28.08.15. [Online]. Available: https://www.bro.org

[27] Antidoto. Abgerufen am 29.08.15. [Online]. Available: https://github.com/FastVPSEestiOu/antidoto

[28] J. T. Rødfoss, "Comparison of open source network intrusion detection systems," 2011.

[29] BotFlex. Abgerufen am 15.09.2015. [Online]. Available: https://github.com/sheharbano/BotFlex