TU Kaiserslautern & DFKI
Image Understanding and Pattern Recognition
Prof. Dr. Thomas Breuel

Projektarbeit

Klassifikation von HTTP Anfragen durch Zeitreihenanalyse

Christian Jansohn

7. April 2008

Betreuer:
Markus Goldstein
&
Prof. Dr. Thomas Breuel

Abstract

Webseiten im Internet werden immer wieder Opfer von böswilligen Angriffen, die das Ziel haben, das komplette Angebot der Seite lahmzulegen. Eine Art von Angriffen sind Distributed Denial of Service (DDoS) Angriffe, die eine immer größere Bedrohung werden. Da die herkömmlichen Varianten durch z.B. UDP und TCP SYN Floods immer effektiver abgewehrt werden, konzentrieren sich neue Varianten auf die Applikationsebene. Zusätzlich ist es schwierig, hochverteilte Angriffe durch einfache Schwellenwerte abzuwehren. Daher ist ein mächtigerer Ansatz zur Erkennung von Angriffen vonnöten. Der in dieser Arbeit vorgestellte Ansatz basiert auf der Annahme, dass sich die Zugriffszeiten der HTTP Anfragen von menschlichen Benutzern auf eine Webseite von denen der Angriffe unterscheiden. Um dies für eine Klassifikation verwenden zu können, wird ein hierarchischer Ansatz zur Modellierung der HTTP Anfragen einer IP Adresse auf eine Webseite verwendet. Auf der unteren Stufe werden einfache Zugriffe modelliert. Auf der nächsten Ebene werden Klicks während eines Besuches auf eine Webseite modelliert und auf der oberen Ebene schließlich wiederkehrende Besuche auf die Seite. Die Zugriffe werden hierbei als Zeitreihe aufgefasst. Dies ermöglicht es, bekannte Methoden zur Zeitreihenanalyse anzuwenden, die Parameter für eine anschließende Klassifikation liefern. Die betrachteten Methoden zur Zeitreihenanalyse sind zum einen ein Modell aus Erwartungswert und dazugehöriger Varianz und zum anderen die bekannten ARIMA Modelle. Für die Klassifikation wird ein Entscheidungsbaum verwendet. Die Zugriffszeiten stammen aus einer HTTP Logdatei, welche in der Regel von Webservern eingesetzt wird. Die Ergebnisse zeigen, dass man die HTTP Zugriffszeiten mit dem vorgestellten hierarchischen Ansatz modellieren kann. Insbesondere eignen sich die untere und die mittlere Ebene dazu, die Angriffe von menschlich erzeugten Zugriffen zu unterscheiden. Auf der oberen Ebene lassen sich zustzlich noch automatisierte Zugriffe von Bots von den menschlichen Zugriffen unterscheiden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Das Internet ist in den vergangenen Jahren ein wichtiger Bestandteil des Lebens vieler Menschen geworden. Viele Menschen verwenden das Internet täglich, um z.B. Nachrichten abzurufen oder um Einkäufe bequem und einfach von zuhause aus zu erledigen. Mit der Zunahme an menschlichen Benutzern erhöhen sich aber auch die Anzahl der automatisierten Zugriffe von Bots, die z.B. Suchmaschinen zur Indizierung von Webseiten verwenden. Es treten auch vermehrt bösartige Angriffe auf, die das Ziel haben, komplette Seiten lahmzulegen, um so z.B. den Betreibern der Seite Schaden zuzufügen. Falls eine Webseite, die hauptsächlich zum Handeln verwendet wird, durch einen Angriff eine gewisse Zeitspanne nicht verfügbar ist, entsteht dem Besitzer ein immenser wirtschaftlicher Schaden, da in dieser Zeitspanne keine Geschäfte abgeschlossen werden können. Wenn die Seite über einen großen Zeitraum hinweg nicht verfügbar ist, wandern Kunden ab, und es entsteht zusätzlich ein hoher Imageverlust.

Eine Art von Angriffen, deren Ziel es ist, eine komplette Webseite lahmzulegen, sind *Distributed Denial of Service*(DDoS) Angriffe [11]. Diese Angriffe nutzen aus, dass die Ressourcen des Servers, der die Webseite verwaltet, begrenzt sind. Wenn sehr viele Anfragen zur gleichen Zeit getätigt werden, kann der Server mit der Menge der Anfragen nicht umgehen. Der Server bricht unter der Last der Anfragen zusammen, die Seite ist nicht mehr verfügbar. Um die erforderliche große Anzahl simultaner Zugriffe zu realisieren, werden diese häufig von so genannten Botnetzwerken ausgeführt. Diese Netzwerke bestehen aus vielen gewöhnlichen Heimcomputern, die durch ein Programm, einem eingefangenem Virus aus dem Internet, infiziert wurden. Zum Zeitpunkt des Angriffs wird dieses Programm ausgeführt und startet eine Folge von Anfragen. Jede einzelne dieser Folgen von Anfragen ist für sich nicht gefährlich, das Problem entsteht durch die Masse an Anfragen zum gleichen Zeitpunkt. Zu den herkömmlichen Angriffen mit UDP, ICMP und TCP SYN Floods, für die Gegenmaßnahmen existieren, gesellen sich vermehrt Angriffe auf der Applikationsebene [10]. Diese Angriffe sind schwerer zu erkennen als die herkömmlichen DDoS Angriffe. Es stellt sich also die Frage, wie man einen solchen Angriff auf der Applikationsebene oder auch andere automatisierte Zugriffe auf Webseiten von den Anfrage normaler menschlicher Benutzer unterscheiden kann.

Der in dieser Arbeit vorgestellte Ansatz zur Klassifikation verschiedener Zugriffsarten basiert auf Zeitreihenanalyse. In anderen Arbeiten [1, 7, 16] wurde schon Netzwerktraffic mit Zeitreihen modelliert und vorhergesagt. Im Gegensatz zu diesen Arbeiten werden in diesem Ansatz die HTTP Zugriffszeiten als Basis für die Zeitreihen verwendet und es wird nicht der komplette Verkehr auf eine Seite betrachtet, sondern die Zugriffe einzelner IP Adressen, um einzelne Quellen zu klassifizieren.

Das Ziel ist es, das Benutzerverhalten von dem automatisierten Zugriffsverhalten anhand der Zugriffszeiten zu unterscheiden. Ein menschlicher Internetbenutzer

greift auf eine Webseite meistens mit einem Webbrowser zu. Der Browser lädt die Seite und zeigt sie auf dem Monitor des Benutzers an. Er wird sich dann die Seite anschauen und nach einiger Zeit eine andere Seite aufrufen. Bei einem Bot erwartet man ein anderes Zugriffsverhalten. Entweder wird die Seite komplett innerhalb weniger Sekunden geladen oder es wird nur auf einige wenige Elemente dieser Seite zugegriffen. Insgesamt sollte das Verhalten eines Bots regelmäßiger sein, als das eines Menschen. Ein Benutzer wird aber eventuell über einen größeren Zeitraum hinweg eine Webseite wiederholt aufrufen. Dabei wird der Abstand dieser Besuche eine gewisse Regelmäsigkeit haben, die abhängig vom Benutzer und der Webseite ist. Betrachtet man eine Seite, die Nachrichten anzeigt, so wird derjenige, der die Seite häufig aufruft, sie z.B. morgens, mittags und abends besuchen. Wenn man davon ausgeht, dass dieser Benutzer einen geregelten Tagesablauf hat, dann werden diese Besuche immer ungefähr zur gleichen Zeit stattfinden. Morgens wenn er das Büro betritt, mittags in der Mittagspause und abends, nach der Arbeit, wenn er daheim ist. Zusätzlich wird es Abweichungen davon geben. Der Benutzer wird häufiger auf diese Nachrichtenseite zugreifen, wenn etwas passiert ist, dass ihn sehr interessiert. Auch von einem Bot erwartet man diese wiederkehrende Zugriffe. Allerdings erwartet man bei ihnen eher genau den selben Abstand zwischen den Besuchen, oder auch wieder einen Unterschied in der Anzahl der Zugriffe, im Vergleich zu einem Menschen. Bei Angriffen ist im Gegensatz dazu ein wiederkehrendes Aufrufen über einen größeren Zeitraum hinweg, wie z.B. Tage oder Wochen, eher nicht zu erwarten.

Um diese verschiedenen Arten von Zugriffszeiten und Regelmäsigkeiten modellieren zu können, werden die Zugriffszeiten in drei Hierarchiestufen unterteilt. Die unterste Stufe soll die Zeiten der einzelnen, unmittelbaren Zugriffe modellieren. Die mittlere Stufe modelliert einen Besuch auf einer Webseite. Die obere Stufe soll das wiederkehrende Verhalten mehrerer Besuche modellieren. Dabei ist jede dieser Stufen als Zeitreihe der Zugriffe in einem Zeitintervall zu verstehen. Diese Zeitreihen sollen mit Modellen beschrieben werden, wie z.B. dem ARIMA Modell. Die entstehenden Modellparameter werden anschließend zur Klassifikation mit Entscheidungsbäumen verwendet.

Diese Arbeit besteht aus vier Teilen. In den weiteren Abschnitten der Einleitung werden das *Hypertext Transfer Protocol* (HTTP) und Logdateien vorgestellt. Des weiteren werden die Daten vorgestellt, auf denen die Ergebnisse dieser Arbeit basieren. Im folgenden Kapitel werden Grundlagen von Zeitreihen und die verwendeten Modelle erläutert. Auch werden Entscheidungsbäume kurz vorgestellt und die Hierarchiestufen, in denen die Zugriffszeiten modelliert werden sollen beschrieben. Das dritte Kapitel fasst die Ergebnisse zusammen und das letzte Kapitel gibt einen Ausblick auf weiterführende Forschungsmöglichkeiten.

1.1 HTTP und Logdateien

Zur Klassifikation sollen die Zugriffszeiten auf einen Webserver verwendet werden, die durch HTTP Anfragen auf diesen Server entstehen. Die Anfragen werden in einer Logdatei gespeichert. Eine solche Datei ist die Basis für die Zeitreihenanalyse und die darauf aufbauende Klassifikation. Zum besseren Verständnis der Entstehung der Zugriffszeiten mit HTTP wird das Protokoll in diesem Abschnitt kurz vorgestellt. Weiter wird das Apache Log Format erläutert, da die verwendete Logdatei in diesem Format erzeugt wurde.

1.1.1 HTTP

HTTP steht für *"Hypertext Transfer Protocol"* und ist ein Protokoll für die Übertragung von Daten im Internet. Für den Datenaustausch zwischen Client und Server wird eine TCP Verbindung üblicherweise an Port 80 des Servers geöffnet. Über diese Verbindung werden die HTTP Nachrichten geschickt, die aus einer oder mehreren Zeilen ASCII Text bestehen. HTTP unterstützt mehrere Methoden. Eine Auswahl dieser Methoden ist in Tabelle 1.1 aufgelistet. Die gebräuchlichste dieser Methoden aus Sicht der Webbrowser ist die GET Methode, da sie genutzt wird Ressourcen abzurufen. Ressourcen sind in diesem Zusammenhang z.B. HTML-Seiten, Bilder, Musikdateien, Videodateien u.a. Dem Methodenaufruf folgt der Pfad und der Dateiname der angefragten Datei. Eine einfache Anfrage auf eine Ressource (in diesem Fall die Datei `index.html`) könnte z.B. so aussehen:

```
GET /index.html HTTP/1.1
```

Hinter der jeweilgen Anfrage wird die Protokollversion angegeben. Es wird zwischen den Versionen 1.0 und 1.1 unterschieden. Ein gravierender Unterschied der beiden Versionen ist, dass bei Version 1.0 für jede Anfrage eine eigene TCP Verbindung geöffnet wird, während bei Version 1.1 mehrere HTTP Nachrichten in einer TCP Verbindung übertragen werden können.

Auf eine Anfrage des Clients antwortet der Server wieder mit einer HTTP Nachricht. Diese enthält einen Statuscode, der angibt ob die Anfrage erfolgreich war oder nicht. Es gibt fünf verschiedene Gruppen von Statuscodes, wobei jeder Code aus drei Ziffern besteht. Die erste Ziffer dient zur Unterscheidung der jeweiligen Gruppe. Der genaue Code wird durch die weiteren Ziffern bestimmt. Eine Übersicht über die verschiedenen Codegruppen kann man in Tabelle 1.2 finden. In der ersten Spalte ist der Code angegeben, in der zweiten Spalte eine Beschreibung der jeweiligen Gruppe und in der letzten Spalte ein konkretes Beispiel.

Tabelle 1.1: Auswahl einiger HTTP Methoden

Methode	**Beschreibung**
GET	Anfrage eine Ressource zu lesen
HEAD	Anfrage Metainformation der Ressource zu lesen
PUT	Legt ein Objekt auf dem Server ab
POST	Fügt Daten auf einer Seite hinzu
DELETE	Löscht die Ressource

Tabelle 1.2: Ein Überblick über die verschiedenen Statuscodegruppen.

Code	**Bedeutung**	**Beispiel**
1xx	Infomation	100 = Server behandelt die Anfrage des Clients
2xx	Erfolg	200 = Die Anfrage war erfolgreich
3xx	Weiterleitung	301 = Die Seite ist umgezogen
4xx	Fehler des Clients	404 = Ressource nicht gefunden
5xx	Fehler des Servers	500 = interner Server Fehler

Die HTTP Nachricht kann noch durch so genannte *Header* erweitert werden, um zusätzliche Informationen zu übermitteln. Dabei wird zwischen *Request* und *Response* Header unterschieden. Ein *Request* Header wird von dem Client verwendet. Mögliche Request Header sind z.B *User-Agent*, *Accept* und *Referer*. Bei einem

Browser wird mit User-Agent die Browserkennung bei einer Anfrage angegeben. Accept gibt die Art von Objekten an, die der Clientbrowser akzeptiert und der Referer ist die URL, die der Browser vorher besucht hat. Ein Response Header erweitert die HTTP Antwort des Servers, so kann hiermit z.B. der Pfad zu einer verschobenen Seite angegeben werden [15]. Weitere Informationen über HTTP, sind in der RFC 1945 [5] (HTTP 1.0) und der RFC 2616 [6] (HTTP 1.1) zu finden.

1.1.2 Das Apache Log Format

Alle HTTP Zugriffe auf einen Server werden in der Regel gesammelt und in einer Datei, einem sogenannten Log, gespeichert. Dies ermöglicht es, z.B. Benutzerverhalten zu analysieren oder Fehler zu finden. Solche Logdateien können unterschiedliche Formate haben. Eines dieser Formate ist das *Apache* Format [13], welches bei Apache Webservern verwendet wird. Da es weit verbreitet ist und da ein solches Log die Zugriffszeiten enthält, die zur Analyse benötigt werden, wird es in diesem Abschnitt kurz vorgestellt.

Eine Zeile in einer Logdatei entspricht einer HTTP Anfrage auf den Server. Es gibt mehrere Komponente, die in einer Zeile aufgelistet werden. Ein Beispiel eines Zugriffes, wie er in einem Log gespeichert wird, ist: `154.168.23.42 - - [22/Sep/2004:08:15:00 +0100] ‘‘GET /javascript/showimages.js HTTP/1.1’’ 200 1513 ‘‘http://www.xvid.org/’’ ‘‘Mozilla/5.0(Windows; U; Windows NT 5.1; en-US; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1’’`

Nach diesem Format werden die folgenden Komponente gespeichert:

- Die IP Adresse von der die Anfrage getätigt wurde, in diesem Fall erfolgte der Zugriff von der IP Adresse `154.168.23.42`.
- Die Identifikation des Clients nach dem *Identification Protocol* RFC 1413 [4]. Sie ist unzuverlässig und wird deshalb oft weggelassen, wie auch in diesem Fall. Dies wird durch `-` dargestellt.
- Die ID des Benutzers basierend auf der HTTP Authentifizierung, welche in dem Beispiel auch nicht angegeben wurde.
- Der Zeitpunkt an dem der Zugriff erfolgt ist: `22/Sep/2004:08:15:00 +0100`.
- Die eigentliche HTTP Nachricht. In dem Beispiel wurde die Datei `showimages.js` angefragt.
- Der Statuscode den der Server zurückschickt, siehe Tabelle 1.2. Die Anfrage war in diesem Fall also erfolgreich.
- Größe des Objektes, das zurück gegeben wird, im Beispiel 1513 Byte.
- Der Referer, also die Webseite auf der man sich vorher befunden hat. Hier war es die Seite `www.xvid.org`.
- Der User-Agent HTTP Request Header, also die Informationen, die der Browser des Clients über sich sendet.

Viele Internetbenutzer rufen Webseiten mit einem Webbrowser wie z.B. Mozilla Firefox, Safari oder dem Microsoft Internet Explorer auf. Wenn nun eine Webseite mit vielen Elementen wie z.B. Bildern, mit einem Browser angefragt wird, wird zuerst die GET Methode auf die HTML-Seite angewendet. Darauf folgt die Anfrage aller weiterer Elemente dieser Webseite. Dies geschieht in einem sehr kurzen Zeitraum, meist innerhalb weniger Sekunden. Dies hat zur Folge, dass viele Anfragen

der selben IP Adresse innerhalb eines kurzen Zeitraumes in dem Logfile auftauchen. Die genaue Größe des Zeitraums ist von mehreren Dingen abhängig, unter anderem der Anzahl der Elemente auf der Webseite, oder der Geschwindigkeit der Internetverbindung.

Für die Analyse werden nur die Zugriffszeiten benötigt. Diese sind in dem Log enthalten. Allerdings sind die Daten aus der Logdatei nicht gelabelt, d.h. man weiß nicht, ob der Zugriff von einem Benutzer oder von einem Bot verursacht wurde, oder ob der Zugriff Teil eines Angriffes ist. Im folgenden Abschnitt wird deshalb darauf eingegangen, wie man die Daten kennzeichnen kann. Dies wird am Beispiel der Daten vorgestellt, die für diese Arbeit verwendet wurden.

1.2 Beschreibung der Daten

Um das echte Verhalten von Netzwerktraffic zu analysieren, benötigt man echte Zugriffsdaten. Die Daten in dieser Arbeit sind geloggte Zugriffe auf die Seite `www.xvid.org` innerhalb eines Zeitraumes von hundert Tagen, zwischen dem 12. Februar und dem 23. Mai 2006. Ein großes Problem dieser Daten ist, dass nicht bekannt ist, ob sich hinter einer IP Adresse ein menschlicher Benutzer oder ein Bot verbirgt. Um die Daten des Logs verwenden zu können, müssen sie deswegen erst noch gelabelt werden.

Einen Anhaltspunkt für die Identität, die sich hinter einer IP Adresse verbirgt, ist die Browserkennung. Manche Bots geben sich dadurch zu erkennen, z.B. der Googlebot. Er verwendet als Browserkennung einfach ‘‘`Googlebot`’’. Andere Bots haben auch eindeutige Kennungen. Wenn als Browserkennung ein gewöhnlicher Webbrowser angegeben ist, ist dies eher ein Anzeichen für einen Benutzer. Die Browserkennung ist aber nicht eindeutig und kann auch gefälscht werden. So geben sich einige Bots als Browser aus.

Einen weiteren Anhaltspunkt liefern die HTTP Anfragen selbst. So deutet eine Anfrage auf die Datei ‘‘`robots.txt`’’[9] auf einen Bot hin, da in ihr das gewünschte Verhalten der Bots auf eine Seite festgelegt wird. Auch können die Endungen der angefragten Dateien auf die Quelle der Anfrage schließen lassen. Dateien mit den Endungen `.gif`, `.html`, `.htm`, `.jpg` und `.pdf` werden z.B. eher von Benutzern mit Browser abgefragt, als von Bots.

Mit solchen Überlegungen kann man die IP Adressen und ihre Zugriffszeiten aus der Logdatei filtern und Klassen zuordnen. Es wurde dabei zwischen folgenden Klassen unterschieden:

- menschliche *Benutzer*
- gutartige *Bots* wie z.B. Indizierungsbots von Suchmaschinen
- *Angriffe* und bösartige Bots
- *Unbekannt*, also Adressen, die nicht zugeordnet werden konnten

Für die weitere Analyse und Klassifikation sind nicht alle dieser gefundenen IPs nützlich. So wurden alle IP Adressen, die weniger als hundert Zugriffe in dem Log hatten, nicht weiter verwendet. Ein Grund für diese Entscheidung ist, dass die resultierenden Zeitreihen zu kurz werden könnten. Die Adressen, die nicht zugeordnet werden konnten, wurden ebenfalls verworfen.

Das Labeln der Daten wurde in mehreren Schritten realisiert. Der Baum in 1.1 stellt eine Übersicht über das genauere Vorgehen des Labelns dar. Jeder Schritt besteht aus einem Filter, der mit regulären Ausdrücken arbeitet. Jede Zeile wird dabei von dem jeweiligen Ausdruck untersucht und wenn der reguläre Ausdruck die Zeile akzeptiert, hat der Filter Erfolg, ansonsten nicht. Ein Erfolg wird in dem Baum

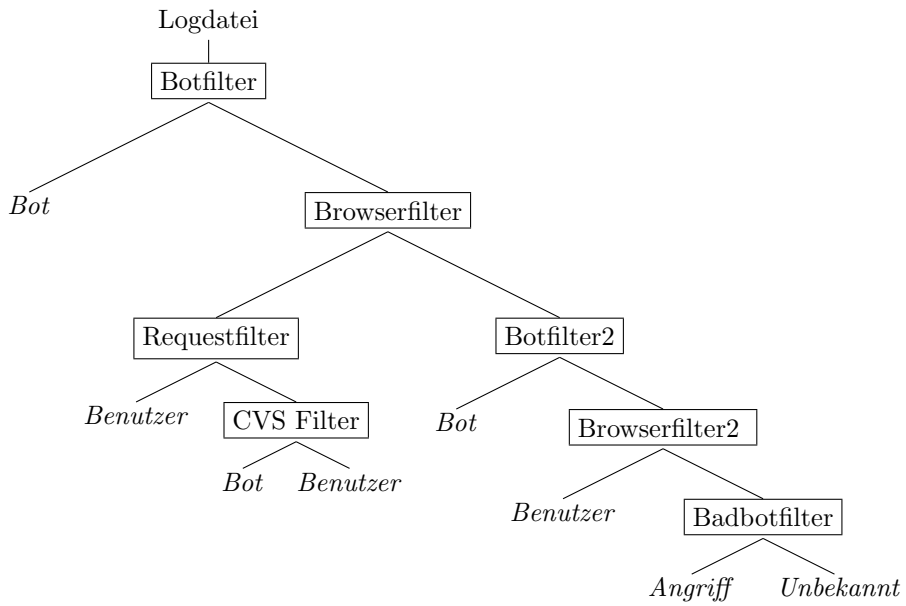

Abbildung 1.1: Übersicht der verschiedenen Filter zum Labeln einer Logdatei.

durch einen Pfad nach links repräsentiert. Der erste Filter prüft nach den offensichtlichsten Bots, wie z.B. dem *Googlebot*, *RSS-Reader*, *Bloglines* oder *Slurp* und die Anfrage der `robots.txt`. Der *Browserfilter* sucht nach gängigen Webbrowsern, wie z.B Firefox, Internet Explorer, Opera und verschiedenen Mobiltelefonbrowsern. Der *Requestfilter* soll überprüfen, ob Dateien mit mit Endungen geladen werden, die auf einen Benutzer schließen lassen. Eine Auswahl dieser Endungen wurde weiter oben schon aufgelistet. Weitere Endungen sind z.B.: `.php`, `.png`, `.tar`, `.txt` oder `.xml`, also Dateien, die Webseiten und Elemente darauf enthalten, bzw. Dateien die häufig herunter geladen werden, wie Musik- oder Videoformate. Falls Dateien aus dem CVS abgerufen werden, wird hier auf einen Bot geschlossen, da dabei sehr viele Dateien innerhalb weniger Sekunden geladen werden und dieses Verhalten weniger dem eines Benutzers entspricht.

Botfilter2 und *Browserfilter2* dienen dazu, die weniger offensichtlichen Bots und Anfragen zu untersuchen. Da es viele verschieden Bots gibt, gibt es auch viele verschiedene User-Agent Header für Bots. Da nicht alle Header der Bots, die auf die betrachtete Webseite zugreifen, zu Beginn bekannt war, wurden die Botfilter geteilt. Beim Filtern der Anfragen verhält es sich ähnlich. Der *Browserfilter* erkennt nicht alle Anfragen von Benutzern, deshalb wird der zweite Filter verwendet. Die IP Adressen, die bis jetzt noch keiner Klasse zugeordnet wurden, werden mit dem *Badbotfilter* auf unsinnige Anfragen untersucht. Solche Anfragen enthalten, als URLs, bekannte Exploitsignaturen, wie z.B. `‘‘w00t’’`.

Es gab aber nur sehr wenige IP Adressen, die der Klasse *Angriff* zugeordnet werden konnten. Außerdem sind diese Angriffe keine DDoS Angriffe und somit keine Angriffe, die in dieser Arbeit erkannt werden sollen. Die Adressen, die bis jetzt noch nicht zu einer Klasse gezählt wurden, wurden als Unbekannt deklariert. Falls eine IP mehreren Klassen zugeordnet wurde, muss sie einer einzigen Klasse zugeordnet

werden. Die Reihenfolge der Zuordnung ist folgendermaßen: *Bot*, *Benutzer*, *Angriff* und *Unbekannt*. Dies bedeutet, falls eine IP Adresse durch eine Zeile des Logs als Bot identifiziert wurde und durch eine weitere Zeile als Benutzer, so wurde die Adresse als Bot gezählt. Der Grund für diese Ordnung ist, die Annahme, dass Quellen für HTTP Anfragen, die sich als Bots ausgeben, mit einer hohen Wahrscheinlichkeit auch Bots sind.

Ein großes Problem ist, dass IP Adressen nicht unbedingt fest vergeben sind, d.h. eine IP kann von mehreren Benutzern verwendet werden, oder auch zuerst von einem Benutzer und dann wieder von einem Bot und umgekehrt. In diesem Fall macht man an dieser Stelle einen Fehler. Es ist nicht möglich, nur aus dem Log herauszufinden, ob sich hinter einer IP Adresse eindeutig ein Benutzer oder ein Bot verbirgt. Mit diesem Vorgehen ist es aber möglich, die IP Adressen mit einiger Sicherheit diesen Klassen zuzuordnen.

Für die weitere Arbeit wurden die IP Adressen der Klassen Angriff und Unbekannt nicht weiter verwendet. Um nun Zugriffszeiten von Angriffen zu bekommen, wurden zwei verschiedene Klassen von Angriffen selbst generiert. *WGET-Angriff* enthält Zugriffe mit WGET auf die Seite. Mit WGET wird die komplette Webseite mit allen dazugehörenden Elementen rekursiv heruntergeladen, deswegen entstehen damit sehr viele Zugriffe (>100) innerhalb einer sehr kurzer Zeit (<10 Sekunden). *Verteilter Angriff* enthält Zugriffe mit einer normalverteilten Zugriffsrate mit einem Mittwelwert von sechs Sekunden, einer ebenfalls normalverteilten Abweichung davon von ein bis zwei Sekunden und einer festen Länge von 1200 Zugriffen Diese beiden Klassen sollen DDoS Angriffe simulieren.

Die so erhaltenen Zugriffszeiten der IP Adressen der unterschiedlichen Klassen wurde im Verhältnis von 80 zu 20 in Trainings- und Testdaten aufgeteilt. Die genaue Anzahl der Daten der Klassen ist in Tabelle 1.3 aufgelistet. Diese Daten sind die Grundlage für die Modelle, welche im folgenden Kapitel beschrieben werden.

Tabelle 1.3: Übersicht über die Größe der Daten

Klasse	Anz. insgesamt	Anz. Training	Anz. Test
WGET-Angriff	5643	4515	1128
Verteilter Angriff	50000	40000	10000
Benutzer	34105	27285	6820
Bot	1620	1296	324

Kapitel 2

Modelle

In diesem Kapitel werden die Mustererkennungsmodelle vorgestellt, die bei der Klassifizierung verwendet werden. Da die HTTP Zugriffe von einer IP Adresse als Zeitreihe interpretiert werden soll, werden im ersten Abschnitt einige Grundlagen zur Zeitreihenanalyse vorgestellt. Diese Analyse liefert uns Erkennungsmerkmale, die wir zum Klassifizieren verwenden. Für die Klassifikation werden Entscheidungsbäume eingesetzt, welche im zweiten Abschnitt vorgestellt werden. Im dritten Abschnitt wird vorgestellt, wie die Zugriffszeiten modelliert werden, also wie man eine Zeitreihe aus den Zugriffszeiten erzeugt.

2.1 Zeitreihen

Eine *Zeitreihe* $\{y_t\}$ ist eine Reihe, in der einem Zeitpunkt ein Wert zugeordnet wird, also: $\{y_t\}$ mit $t = 1, 2, ..., n$ und $y_t \in \mathbb{R}$ (s. auch [12] und [14]). Im Folgenden wird die Notation aus [14] verwendet. Ein Beispiel einer Zeitreihe findet man in Abbildung 2.1. In dieser Reihe werden die Zugriffe auf `www.xvid.org` pro Tag gezählt. Auffällig sind die regelmäsigen Spitzen des Graphen, die einen Abstand von ca. sieben Tagen haben. An den Wochenenden finden demnach mehr Zugriffe auf die Webseite statt als unter der Woche.

Zur Bestimmung der linearen Abhängigkeit zweier Punkte einer Zeitreihe kann die Autokovarianzfunktion verwendet werden, die wie folgt definiert ist:

$$\gamma_y(s,t) = E[(y_s - \mu_s)(y_t - \mu_t)] \tag{2.1}$$

für alle s, t. Hierbei ist μ_t der Erwartungswert der Zeitreihe zum Zeitpunkt t. Mit der Autokovarianz kann man nun die *Autokorrelationsfunktion* (ACF) bestimmen:

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}} \tag{2.2}$$

Dabei ist zu beachten, dass $-1 \leq \rho(s,t) \leq 1$ gilt. Die ACF ist ein Maß für die lineare Vorhersagbarkeit einer Zeitreihe zum Zeitpunkt t. Falls $y_t = \alpha_0 + \alpha_1 y_s$ gilt, der Wert der Zeitreihe zum Zeitpunkt t also linear von dem Wert der Reihe zum Zeitpunkt s abhängt, so ist $\alpha_1 > 0$, falls die Korrelation gleich 1 ist. Falls die Korrelation den Wert -1 hat, so ist $\alpha_1 < 0$.

Bei Zeitreihen unterscheidet man zwischen *stationären* und *nichtstationären* Zeitreihen. Eine Zeitreihe ist dabei *stationär*, wenn ihr Erwartungswert unabhängig von Zeitpunkt t konstant ist, also wenn gilt:

$$E[y_t] = E[y_{t-s}] = \mu \tag{2.3}$$

und wenn die Kovarianzfunktion $\gamma(s,t)$ von s und t nur durch die Differenz $|s-t|$ bestimmt wird. Diese Differenz wird durch die Zeitverschiebung (das *Lag*) h ausgedrückt, wobei gilt: $s = t + h$. Die Autokovarianzfunktion einer stationären Zeitreihe hat somit folgende Form:

$$\gamma(h) = E[(y_{t+h} - \mu)(y_t - \mu)] \tag{2.4}$$

Daraus ergibt sich für die Autokorrelationsfunktion:

$$\rho(h) = \frac{\gamma(t+h,h)}{\sqrt{\gamma(t+h,t+h)\gamma(t,t)}} = \frac{\gamma(h)}{\gamma(0)} \tag{2.5}$$

Mit der Autokorrelationsfunktion kann man bestimmen, wie stark der Wert der Zeitreihe vom Lag beeinflusst wird. Betrachtet man zwei Zeitreihenwerte, einen zum Zeitpunkt t und den anderen um den Lag h verschoben, so bedeutet ein Korrelationswert der gegen 1 strebt, dass die beiden Werte sehr stark korrelieren. Ist die Korrelation 0, stehen beide Werte in keiner Korrelation zueinander. Ein Wert gegen -1 bedeutet, dass die Werte umgekehrt korrelieren, ist also der eine Wert sehr groß, ist der andere sehr klein. In Abbildung 2.2 ist die ACF zu der Reihe der Zugriffe auf `www.xvid.org` abgebildet. Es ist eindeutig erkennbar, das die Korrelation der Werte bei den Lags mit $h = 6, 7, 8$ und $h = 13, 14, 15$ usw. sehr hoch ist. Das Muster wiederholt sich somit wöchentlich. Dies deutet darauf hin, dass Zahl der Zugriffe vom Wochentag abhängig ist. In Abbildung 2.2 ist auch die *Partielle Autokorrelationsfunktion* (PACF) ϕ_{hh} abgebildet, die für eine stationäre Zeitreihe y_t wie folgt definiert ist:

$$\phi_{11} = corr(y_1, y_0) = \rho(1) \tag{2.6}$$

$$\phi_{hh} = corr(y_h - y_h^{h-1}, y_0 - y_0^{h-1}), h \geq 2 \tag{2.7}$$

Hierbei ist

$$y_h^{h-1} = \beta_1 y_{h-1} + \beta_2 y_{h-2} + ... + \beta_{h-1} y_1 \tag{2.8}$$

die Regression von y_h^{h-1} auf $\{y_{h-1}, y_{h-2}, .., y_1\}$. Analog dazu ist y_0^{h-1} die Regression von y_0 auf $\{y_1, y_2, .., y_{h-1}\}$ und damit:

$$y_0 = \beta_1 y_1 + \beta_2 y_2 + ... + \beta_{h-1} y_{h-1} \tag{2.9}$$

Die PACF drückt die Korrelation zwischen y_t und y_{t-h} aus, wobei die lineare Abhängigkeit zu $\{y_{t-1}, y_{t-2}, .., y_{t-(h-1)}\}$ nicht berücksichtigt wird [14].

2.1.1 Erwartungswert und Varianz

Die im vorigen Abschnitt beschriebenen Funktionen sind theoretische Beschreibungen der Zeitreihen. In der Praxis ist es schwierig, diese Funktionen exakt zu bestimmen, weswegen man sie aus den gegebenen Werten annähert. Um den Erwartungswert zu bestimmen, berechnet man das statistische Mittel:

$$\bar{y} = \frac{1}{n} \sum_{t=1}^{n} y_t = \hat{\mu} \tag{2.10}$$

Die Autokovarianzfunktion lässt sich aus dem Mittelwert wie folgt bestimmen:

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (y_{t+h} - \hat{\mu})(y_t - \hat{\mu}) \tag{2.11}$$

Damit ergibt sich für die ACF:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \tag{2.12}$$

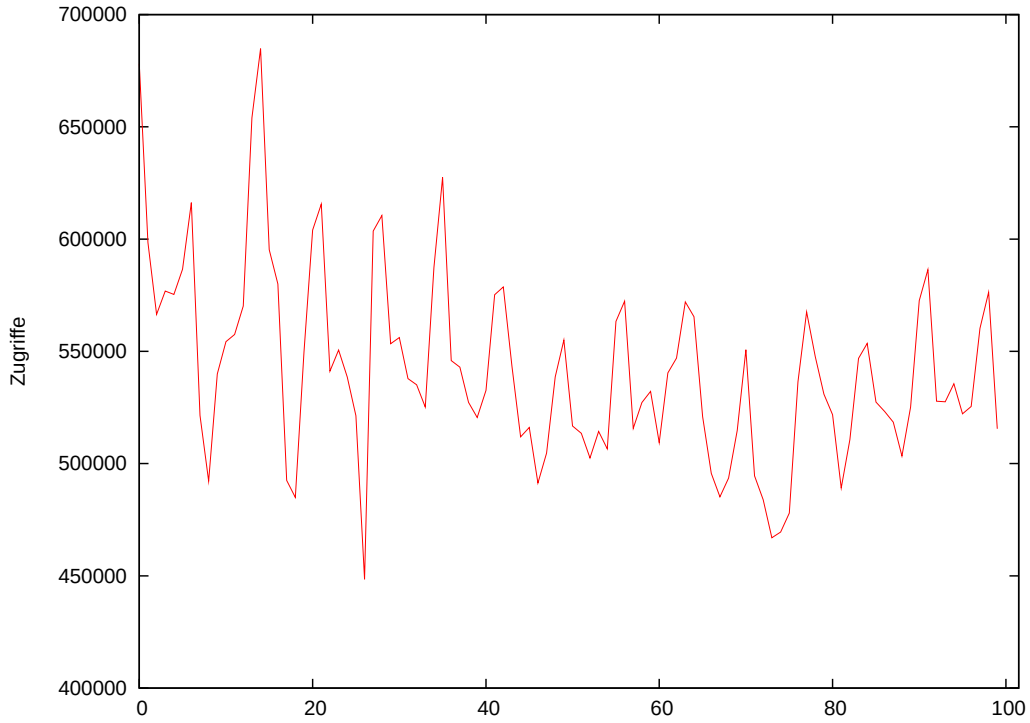

Abbildung 2.1: Zeitreihe über die Anzahle der Zugriffe pro Tag auf `www.xvid.org` zwischen dem 12. Februar 2006 und dem 23. Mai 2006.

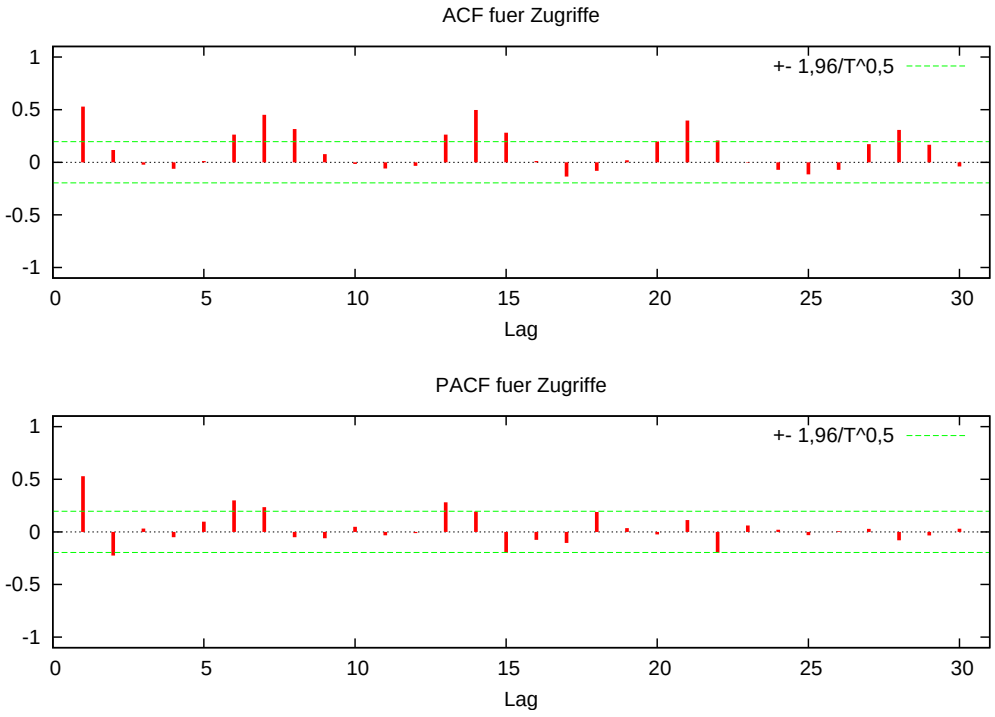

Abbildung 2.2: Die ACF und PACF der Zeitreihe aus 2.1

Die Varianz ergibt sich nun aus der gemittelten quadrierten Abweichung von diesem Mittelwert:

$$\frac{1}{n-1}\sum_{t=1}^{n}(y_t - \hat{\mu})^2 = \hat{\sigma}^2 \tag{2.13}$$

Ein einfaches Modell, um eine Zeitreihe zu beschreiben, ist nun der Mittelwert und die Varianz. Als drittes Feature wird noch die Länge der Zeitreihe verwendet. Es werden zur Beschreibung einer Zeitreihe also die Werte $\hat{\mu}$, $\hat{\sigma}^2$ und n benutzt.

2.1.2 Das ARIMA(p,d,q) Modell

Das ARIMA Modell setzt sich aus drei Teilen zusammen. Der erste Teil ist die *Auto Regression (AR)*. Der zweite Teil ist der *Moving Average (MA)*. Das *I* steht für *Integrated* und beschreibt den dritten Teil. Der *Lagoperator B* ist definiert wie folgt:

$$By_t = y_{t-1} \tag{2.14}$$

Wird der Lagoperator wiederholt angewandt, verhält er sich folgendermaßen:

$$\begin{aligned} B^2y_t &= By_{t-1} = y_{t-2} \\ B^3y_t &= B^2y_{t-1} = By_{t-2} = y_{t-3} \\ &... \\ B^ky_t &= y_{t-k} \end{aligned}$$

Die *Differenz* von Zeitreihenwerten wird wie folgt notiert:

$$\nabla y_t = y_t - y_{t-1} \tag{2.15}$$

Mit Hilfe des Lagoperators aus 2.14 kann man diese Differenz auch so ausdrücken:

$$\nabla y_t = (1-B)y_t \tag{2.16}$$

Somit ist die zweite Differenz

$$\begin{aligned} \nabla^2 y_t &= (1-B)^2 = (1-2B+B^2)y_t \\ &= y_t - 2y_{t-1} + y_{t-2} \end{aligned}$$

Die *Differenz der Ordnung d* ist nun definiert als:

$$\nabla^d = (1-B)^d \tag{2.17}$$

Ein **Autoregressives Model der Ordnung *p* (AR(p))** für eine stationäre Zeitreihe y_t mit Mittelwert 0 hat die Form:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{2.18}$$

wobei $\phi_p \neq 0$ und ϵ_t *Weißes Rauschen (White Noise)* ist. Darunter versteht man einen normalverteilten Prozess mit Mittelwert 0 und Standardabweichung σ.

$$\epsilon_t = \mathcal{N}(0, \sigma^2) \tag{2.19}$$

Für die Autokorrelation des Weißen Rauschens gilt: $\rho_0 = 1$, $\rho_1 = \rho_2 = ... = 0$, die Werte korrelieren also nicht. Bei AR(p) drückt man den Wert zum Zeitpunkt t durch eine gewichtete Summe der vorherigen p Zeitreihenwerte und einem Fehler, der durch das Rauschen ausgedrückt wird, aus. Ein AR(p) Prozess kann auch mit dem *autoregressiven Operator* $\phi(B)$ formuliert werden:

$$\phi(B)y_t = \epsilon_t \tag{2.20}$$

wobei

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p \tag{2.21}$$

ist.

Falls der Mittelwert μ der Zeitreihe nicht 0 ist, kann man eine Zeitreihe mit 0 als Mittelwert erhalten, indem man μ von jedem Wert y_t abzieht. Auf diese resultierende Zeitreihe kann man das AR Modell anwenden.

Das **Moving Average Modell der Ordnung *q* (MA(q))** einer Zeitreihe y_t ist definiert als:

$$y_t = \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - ... - \theta_q \epsilon_{t-q} + \epsilon_t \tag{2.22}$$

wobei $\theta_p \neq 0$ und ϵ_t wieder weißes Rauschen darstellt. Analog zu dem autoregressiven Modell wird der *Moving Average Operator* $\theta(B)$ formuliert:

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + ... + \theta_q B^q \tag{2.23}$$

und somit kann man einen MA(q) Prozess in der Form

$$y_t = \theta(B)\epsilon_t \tag{2.24}$$

darstellen. Bei diesem Modell hängt der Wert zum Zeitpunkt t linear von den q vorherigen Werten des Rauschens ab. Das **Autoregressive Moving Average Modell (ARMA(p,q))** ist die Kombination von AR(p) und MA(q) und hat für stationäre Zeitreihen die folgende Form:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - ... - \theta_q \epsilon_{t-q} \tag{2.25}$$

mit $\phi_p, \theta_q \neq 0$.

Mit den Operatoren aus 2.20 und 2.23 vereinfacht sich diese Gleichung zu:

$$\phi(B)y_t = \theta(B)\epsilon_t \tag{2.26}$$

Das ARMA(p,q) Modell ist nur bei stationären Zeitreihen anwendbar, allerdings sind nicht alle Zeitreihen, die man analysieren will, stationär. Diese nichtstationären Zeitreihen kann man als Zusammensetzung zweier Reihen betrachten, einer nichtstationären Komponente, dem Trend und einer stationären Komponente. Sei z.B. $x_t = \mu_t + y_t$ eine nichtstationäre Zeitreihe, wobei y_t stationär und $\mu_t = \beta_0 + \beta_1 t$ der nichstationäre Trend ist. Die Differenz

$$\nabla x_t = x_t - x_{t-1} = \beta_1 + y_t - y_{t-1} = \beta_1 + \nabla y_t$$

ist hingegen stationär [14]. Je nach Form des Trends der nichtstationären Zeitreihen muss man d mal Differenzieren bevor man eine stationäre Reihe erhält, auf welche man ein ARMA(p,q) Modell anwenden kann. Dieses Modell heißt **Auto Regression Integrated Moving Average Modell der Ordnung *p,d,q* (ARIMA(p,d,q))** und hat die Form:

$$\phi(B)(1 - B)^d y_t = \theta(B)\epsilon_t \tag{2.27}$$

wobei $(1 - B)^d$ die Differenz der Ordnung d ist (s. 2.16).

Zur Klassifikation der Zeitreihen sollen die Koeffizienten ϕ_t und θ_t verwendet werden. Diese können bei gegebenen Parametern p, d und q z.B. mit der *Yule-Walker Estimation* [14] bestimmt werden. Die Herausforderung ist nun, zu einer

gegebenen Zeitreihe das passende ARIMA(p,d,q) Modell, also die Modellparameter p, d und q, zu finden.

Ein Ansatz diese Parameter zu bestimmen, ist der *Box Jenkins* Ansatz. Dieser Ansatz besteht aus mehreren Schritten. Zunächst wird die Zeitreihe geplottet und der Graph untersucht. Dabei wird unter anderem geprüft, ob die Reihe stationär ist, oder ob sich ein erkennbarer Trend abzeichnet. Falls die Reihe nichtstationär ist, kann man die Differenzen bilden und dann die resultierende Reihe untersuchen. Im nächsten Schritt werden die ACF und PACF untersucht, die Rückschlüsse auf die Ordnung eines ARMA Modells geben (siehe auch [1, 8, 14]). So deutet sich ein MA(q) Prozess dadurch an, dass keine signifikanten Spitzen bei der ACF nach Lag q folgen. Falls die PACF nach Lag p keine Spitzen anzeigt, deutet dies auf einen AR(p) Prozess.

Wenn die Ordnung des Modells feststeht, werden die Koeffizienten dieses Modells berechnet und anschließend das Ergebnis auf seine Güte überprüft. Hierzu kann man z.B. die mit dem Modell beschriebene Zeitreihe mit der tatsächlichen Zeitreihe vergleichen, indem man aus den ersten T Zeitpunkten das Modell bestimmt und dann die restlichen $n-T$ Werte mit dem Modell vorhersagt. Ist die Vorhersage gut, hat man ein passendes Modell gefunden, sonst nicht.

Dieser Ansatz ist für unsere Aufgabe eher ungeeignet, da für jede Zeitreihe einzeln die Parameter bestimmt werden müssten, die zu stark auf die jeweilige Zeitreihe spezialisieren. Daher verwendet man andere Techniken, um die Güte des Modells abzuschätzen. Eine dieser Techniken ist *Akaike's Information Criterion* (AIC) [2, 14]:

$$AIC = \ln \hat{\sigma}_k^2 + \frac{n + 2k}{n} \tag{2.28}$$

Hierbei sind $\ln \hat{\sigma}_k^2$ ein Maximum Likelihood Estimator für die Fehler der Modellparameter, k die Anzahl der Parameter und n die Anzahl der gegebenen Zeitpunkte. Das Modell, für welches das AIC minimal ist, soll das beste Modell für die jeweiligen Datensätze sein. Man kann somit Modelle untereinander vergleichen und abschätzen welches besser ist. Im Allgemeinen werden Modelle mit wenigen Parametern dabei bevorzugt. Allerdings kann es passieren, dass, wenn viele Beispieldaten vorhanden sind, ein Modell mit mehr Parametern bevorzugt wird, obwohl ein einfacheres Modell besser wäre. Deswegen wird mit dem *Bayesian Information Criterion* (BIC) ein weiteres Kriterium vorgestellt:

$$BIC = \ln \hat{\sigma}_k^2 + \frac{k \ln n}{n} \tag{2.29}$$

Bei BIC wird eine höhere Anzahl Parameter stärker bestraft, wenn die Anzahl der gegebenen Zeitpunkte größer ist. Um nun ein Modell zu bestimmen, vergleicht man mehrere ARIMA Modelle und nimmt das, das den kleinsten AIC bzw. BIC hat. Als Beispiel sind in Tabelle 2.1 verschiedene ARIMA Modelle zu der Zeitreihe der Zugriffe auf `www.xvid.org` mit den dazugehörigen AIC und BIC Werten. Das ARIMA(4,1,3) Modell hat den minimalen Werte für AIC, also würde man nach diesem Kriterium dieses Modell bevorzugen. Das ARIMA (4,1,2) Modell hat allerdings den minimalen Wert für BIC. Zum Vergleich der beiden Modelle sind in Abbildungen 2.3 (ARIMA(4,1,2)) und 2.4 (ARIMA(4,1,3)) die Graphen der tatsächlichen Zeitreihe und der jeweiligen angepassten Modelle abgebildet.

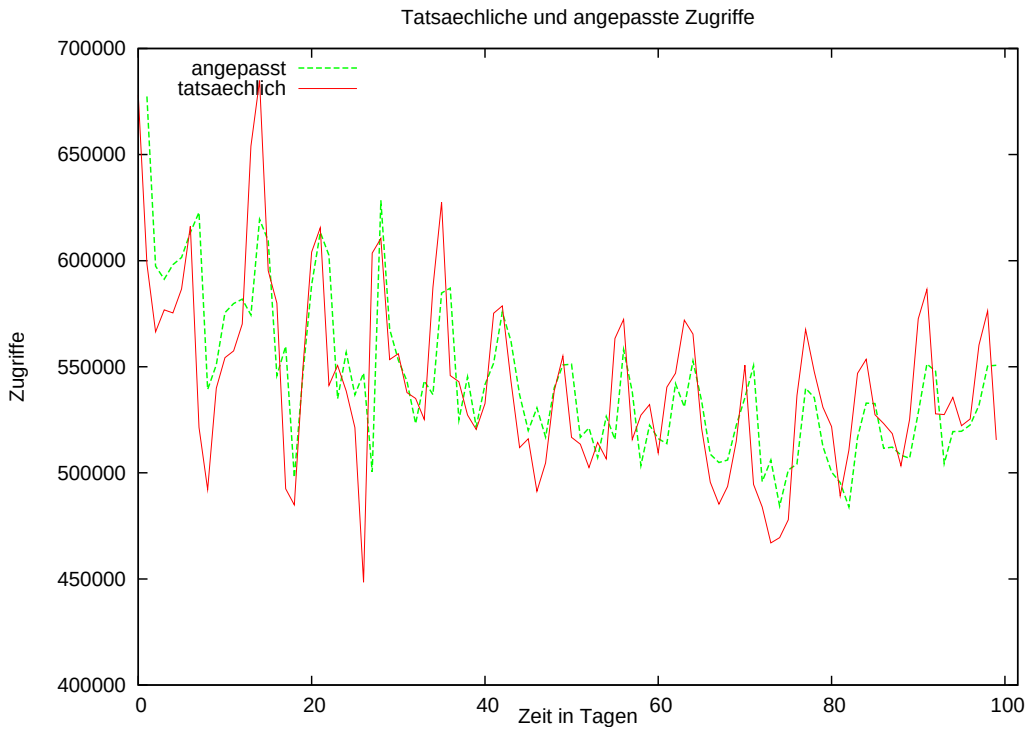

Abbildung 2.3: Die Zeitreihe aus 2.1 mit angepasstem ARIMA(4,1,2) Modell

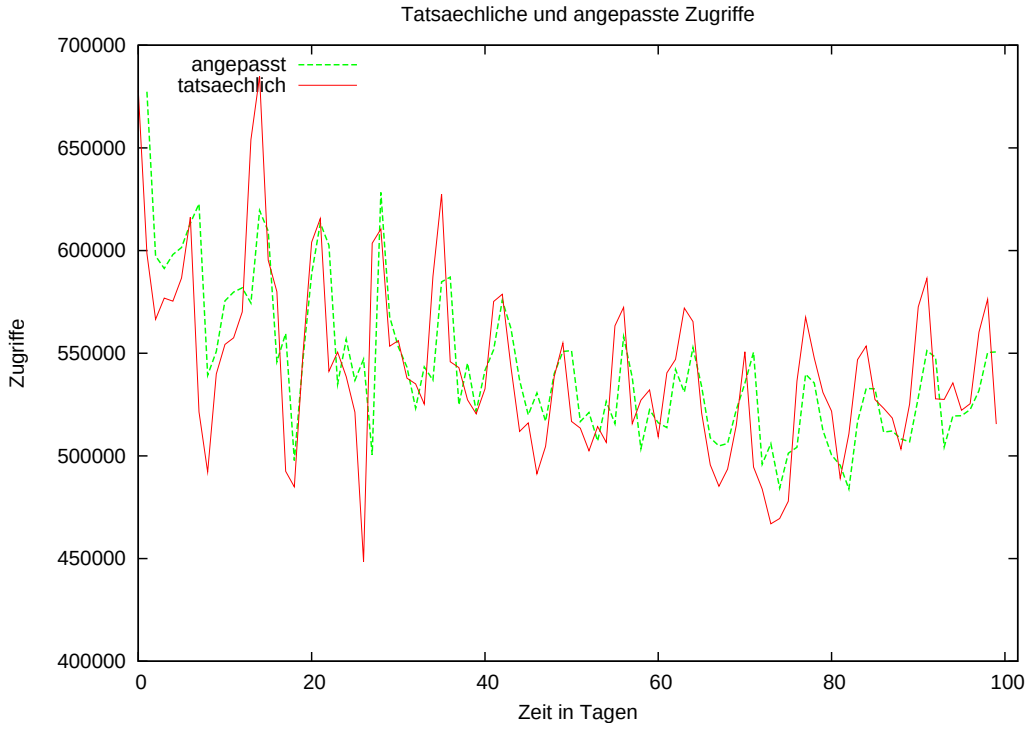

Abbildung 2.4: Die Zeitreihe aus 2.1 mit angepasstem ARIMA(4,1,3) Modell

Tabelle 2.1: AIC und BIC für verschiedene ARIMA Modelle für die Zeitreihe aus 2.1.

ARIMA(p,d,q)	AIC	BIC
ARIMA(1,0,0)	2388.04	2395.86
ARIMA(2,0,0)	2384.4	2394.82
ARIMA(3,0,0)	2386.63	2399.66
ARIMA(0,0,1)	2384.38	2392.19
ARIMA(0,0,2)	2383.73	2394.15
ARIMA(0,0,3)	2385.72	2398.75
ARIMA(1,0,1)	2390.04	2400.46
ARIMA(1,1,1)	2379.34	2389.72
ARIMA(2,1,1)	2359.23	2372.2
ARIMA(3,1,1)	2373.86	2389.43
ARIMA(2,1,2)	2355.36	2370.93
ARIMA(2,2,2)	2367.82	2383.33
ARIMA(3,1,3)	2357.02	2377.78
ARIMA(4,1,2)	2343.69	**2364.45**
ARIMA(4,1,3)	**2343.25**	2366.6
ARIMA(4,2,3)	2369.59	2392.85
ARIMA(4,1,4)	2345.08	2371.03

2.2 Entscheidungsbäume

Mit den erwähnten Modellen lassen sich nun Parameter der Zeitreihen extrahieren. Mit diesen Merkmalen soll nun klassifiziert werden, d.h. es soll bestimmt werden, auf welche der Klassen Benutzer, Bot oder Angriff die Reihe schließen lässt. Um dies zu realisieren, werden Entscheidungsbäume verwendet. Was sich hinter einem Entscheidungsbaum verbirgt, wie er aufgebaut wird und wie man damit klassifizieren kann, wird in diesem Abschnitt erläutert.

Ein Entscheidungsbaum ist ein Baum, bei dem an einem Knoten Eigenschaften der Merkmale der Daten geprüft werden. Abhängig von dieser Eigenschaft wird entschieden, welcher Pfad an diesem Knoten eingeschlagen wird. An den Blättern wird schließlich festgelegt, zu welcher Klasse der Datensatz gehört. Die Klassifikation mit einem Entscheidungsbaum hat den Vorteil, dass Regeln verständlicher für Menschen sind als abstrakte Werte aus Blackbox-Klassifikatoren, wie z.B. künstlichen neuronalen Netzen. Die Eigenschaften des zu klassifizierenden Objektes werden, beginnend bei der Wurzel, an den Knoten überprüft, bis man an einem Blatt angelangt. Dieses Blatt legt die Klasse des Objektes fest. Generell sind beliebige Bäume denkbar, da man aber jeden Baum in einen binären Baum, also einen Baum, bei dem jeder Knoten maximal zwei Kinder hat, umformen kann, werden im Folgenden nur binäre Baume betrachtet [3]. Ein Beispiel eines Entscheidungsbaumes ist in Abbildung 2.5 dargestellt. Dieser Baum entstand durch das Trainieren mit ARIMA(3,1,3) Parametern für die Zugriffszeiten in der oberen Ebene (s. Abschnitt 2.3) auf `www.xvid.org`. Die Wurzel prüft den Wert ϵ, der das Weiße Rauschen darstellt. Ist er kleiner gleich 4.6099, wird ein Benutzer klassifiziert. Falls dies nicht der Fall ist, wird θ_1 geprüft, usw.

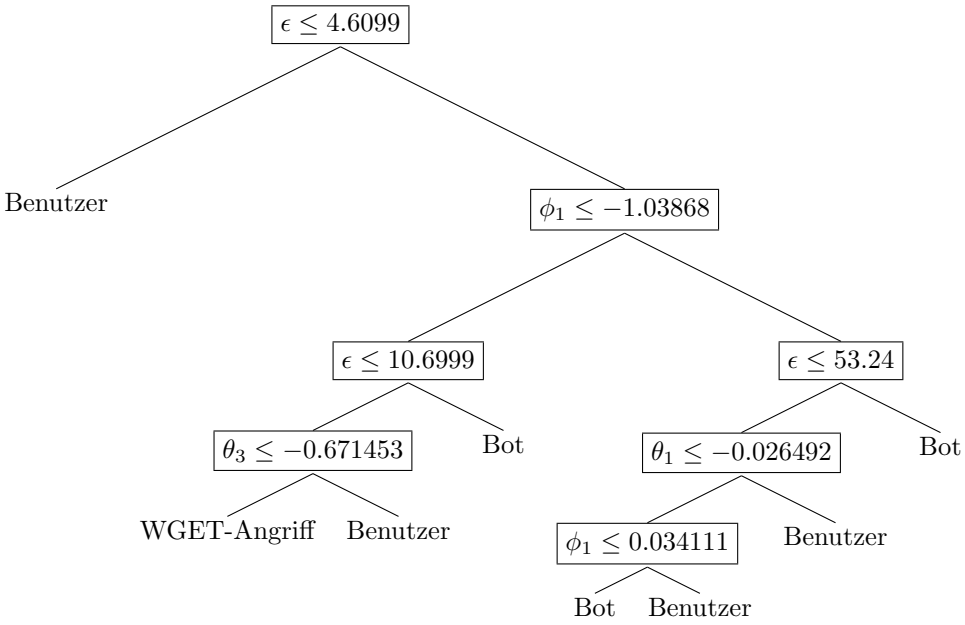

Abbildung 2.5: Entscheidungsbaum zur Klassifikation der Zugriffe der oberen Ebene mit ARIMA(3,1,3)

Zum Aufbau benutzt man eine Menge von Trainingsdaten, die an der Wurzel in disjunkte Teilmengen aufgeteilt wird. Diese Teilmengen werden wieder aufgeteilt, usw. bis nur noch die Blätter übrig bleiben. Das Ziel ist es, Teilmengen zu erhalten, deren Elemente möglichst nur einer Klasse zugeordnet werden können. Für das Training müssen folgende Punkte geklärt werden:

1. Welche Eigenschaft wird an einem Knoten getestet?
2. Wann wird ein Knoten zu einem Blatt, also wann wird die Teilmenge nicht mehr aufgeteilt?
3. Wie kann ein Baum, falls er zu groß (speziell) wurde, wieder gestutzt (generalisiert) werden?
4. Welches Label bekommt ein Blatt, falls nicht alle Elemente der resultierenden Teilmenge zu einer Klasse gehören, das Blatt also nicht rein ist?

Das Ziel des Trainings eines Entscheidungsbaumes ist es, einen Baum zu erhalten, der gute Klassifikationsraten hat. Der Baum soll aber auch möglichst klein sein und entsprechend wenig Knoten haben, um Overfitting zu vermeiden. Aus diesem Grund soll die Entscheidung T (z.B der Form $x_j \leq x_s$) an einem Knoten N Teilmengen erzeugen, die möglichst "rein" sind, d.h. möglichst nur Elemente aus einer Klasse enthalten. Um dies zu messen, wird die *impurity*, die "Unreinheit" eines Knotens $i(N)$ definiert, die 0 sein soll, falls Teilmengen rein sind und groß, falls alle Labels gleich oft in den Mengen repräsentiert werden. Ein mögliches Maß ist die so genannte *entropy impurity*:

$$i(N) = -\sum_j P(\omega_j) log_2 P(\omega_j) \tag{2.30}$$

Hierbei ist $P(\omega_j)$ der Anteil der Elemente an Knoten N der zur Klasse ω_j gehört. Wenn man nun einen Teilbaum bis Knoten N erzeugt hat, wählt man nun die Entscheidung, die die Abnahme der *impurity* $\triangle i$ maximiert. Dabei ist $\triangle i$ folgendermaßen definiert:

$$\triangle i = i(N) - P_L i(N_L) - (1 - P_L) i(N_r) \tag{2.31}$$

Dabei sind N_L und N_R die linken bzw. rechten Nachfolgeknoten, $i(N_L)$, $i(N_R)$ die jeweiligen *impurities* und P_L der Anteil, der Elemente der an den linken Nachfolgeknoten geht. Diese Optimierung ist lokal und hat keine Garantie, dass das globale beste Ergebnis gefunden wird, was bedeutet, dass nach dem Training nicht der kleinste mögliche Baum gefunden wurde [3]. Allerdings kann man die Menge immer weiter aufteilen und dadurch die *impurity* weiter verringern.

Das nächste Problem ist, wann man mit dem Teilen aufhört. Man kann natürlich die Mengen immer weiter verkleinern, bis man nur noch einelementige Mengen hat. Diese haben dann eine *impurity* von 0, allerdings ist dann der Klassifikator auf die Trainingsdaten spezialisiert und entspricht mehr oder weniger einer Tabelle. Testdaten können dann nicht gut erkannt werden (Overfitting). Wenn der Baum aber zu früh mit Aufteilen aufhört, ist die Performanz auf den Trainingsdaten schon gering. Eine Möglichkeit herauszufinden, wann man das Aufteilen beenden kann, ist Validation bzw. Cross-Validation. Bei der Validation werden die Trainingsdaten aufgeteilt und ein Teil (z.B. 90%) zum Trainieren, der restliche Teil (10%) zum Validieren verwendet. Es wird dann solange gesplittet, bis der Fehler auf der Validierungsmenge minimiert wurde. Bei Cross-Validation verwendet man mehrere unterschiedliche Validierungsmengen.

Eine andere Art Entscheidungsbäume zu minimieren, ist das so genannte *Pruning* (Zurückstutzen). Bei dem bisher beschriebenen Vorgehen wird die Entscheidung für ein Aufteilen an einem Knoten nur an dem aktuellen Knoten und nicht anhand der Entscheidungen an den resultierenden Nachfolgeknoten getroffen. Somit kann es passieren, dass ein Knoten als Blatt deklariert wird, obwohl ein weiteres Aufteilen den Erkennungsprozess verbessert hätte.

Beim *Pruning* wird ein Baum erst so lange erzeugt, bis in jedem Blatt die minimale *impurity* herrscht. Nun werden benachbarte Blätter zusammengefasst, falls das Zusammenfassen die *impurity* nur minimal erhöht, d.h. der Elterknoten der beiden Blätter wird selbst zum Blatt, die beiden Blätter fallen weg. Dies wird so lange fortgeführt, bis die Unreinheitswerte zu groß werden.

Das Festlegen des Labels eines Blattes ist einfach. Falls alle Muster eines Blattes zu einer Klasse gehören, bekommt das Blatt das Label dieser Klasse. Sonst wird als Label die Klasse gewählt, die am häufigsten repräsentiert wird. Zusätzlich ist noch zu beachten, dass eine sehr kleine *impurity*, also eine *impurity* von oder sehr nahe bei Null nicht unbedingt notwendig ist, sondern sogar ein Anzeichen von Overfitting sein kann.

2.3 Hierarchische Modellierung der Zugriffszeiten mit Zeitreihen

Im ersten Abschnitt dieses Kapitels wurde vorgestellt, was eine Zeitreihe ist und wie sie analysiert werden kann. Die Ergebnisse dieser Analyse werden als Erkennungsmerkmale für einen Klassifikator verwendet, wie in diesem Fall einem Entscheidungsbaum. In diesem Abschnitt wird vorgestellt, wie die Zeitreihen erzeugt werden, die die Zugriffszeiten auf eine Webseite repräsentieren.

Aus der Logdatei bekommt man die Zugriffszeiten der IP Adressen. Diese Zeiten könnte man direkt als Zeitreihe interpretieren und analysieren, was aber einige Nachteile hat.

Für eine Zeitreihe benötigt man ein Zeitintervall, in dem Werte betrachtet werden. Da die Zeiteinheit bei einem Apache Log eine Sekunde ist, hätte diese Wahl zur Folge, dass die Reihen zum einen sehr lang werden, da die Logdatei einen Zeitraum von hundert Tagen umfasst. Ein weiteres Problem ist, dass viele dieser Werte Null sind, da es häufig passiert, dass eine IP Adresse mehrere Tage nicht auf die Seite zugreift. Unter diesen Bedingungen ist es schwierig, ein ARIMA(p,d,q) Modell auf die Zeitreihen anzupassen und verlässliche Aussagen zu bekommen. Auch das Betrachten des Mittelwertes einer so erzeugten Reihe wäre nicht aussagekräftig, da der Mittelwert bei allen Reihen nahe bei Null wäre. Das Ziel ist es also, eine Zeitreihenrepräsentation zu finden, die das Zugriffsverhalten einer IP Adresse gut beschreibt, dabei aber nicht zu lange wird und nicht zu viele Nullen enthält.

Um auf eine solche Repräsentation zu kommen, betrachtet man das Zugriffsverhalten eines normalen Benutzers. Ein solcher Benutzer verwendet einen Webbrowser, um sich eine Seite im Internet anzuschauen. Er ruft die Seite auf, liest, was ihn interessiert, und ruft dann anschließend einen Link auf der Seite auf, oder er verlässt die Seite wieder, falls er nichts Interessantes mehr gefunden hat. Schließlich ruft er die Seite vielleicht nach einem größeren Zeitintervall (ein paar Stunden, Tagen, eventuell Wochen) wieder auf. Es bietet sich also an, die folgenden drei Stufen des Zugreifens auf eine Seite zu unterscheiden:

1. der erste Zugriff und das damit verbunde Laden der Seitenelemente
2. der Besuch auf der Seite, also das Laden einer Folge von Seiten von dem Server
3. der wiederholte Besuch der Seite über einen größeren Zeitraum hinweg

Mit diesen drei Stufen kommt man auf ein hierarchisches Modell der Zugriffszeiten einer IP Adresse. Diese drei Hierarchiestufen werden im Folgenden genauer beschrieben.

2.3.1 Die untere Hierarchiestufe - die Zugriffe

Diese Stufe soll das Laden aller Elemente beim ersten Aufruf auf die Seite beschreiben. Betrachtet man wieder das erwartete Verhalten eines Benutzers auf eine Webseite, dann besteht der erste Schritt darin, die Webseite mit einem Browser aufzurufen. Der Browser lädt daraufhin alle Elemente dieser Seite und zeigt sie an. In der Logdatei wird das Laden jedes dieser Elemente als einzelner Zugriff gezählt. Somit sind viele Zugriffe dieser IP Adresse in einem sehr kurzen Zeitraum aufgelistet. Der Benutzer wird sich nun die Seite anschauen, was etwas Zeit in Anspruch nimmt, in der keine oder nur sehr wenige Zugriffe (falls z.B. Elemente automatisch nachgeladen werden) auf den Server geloggt werden. Um dieses Verhalten gut Modellieren zu können, wählt man nun ein Zeitintervall, in dem die Zugriffe gezählt werden. Dies soll auf dieser Hierarchieebene die kleinste mögliche Einheit sein, also eine Sekunde. Des weiteren definieren wir einen Timeout von 30 Sekunden. Das bedeutet, dass wenn innerhalb 30 Sekunden zum letzten Zugriff kein weiterer Zugriff erfolgt die Zeitreihe beendet wird. In Abbildung 2.6 ist eine Zeitreihe eines Benutzers, der auf `www.xvid.org` zugreift dargestellt. Man kann dabei gut erkennen, dass in den ersten Sekunden eine große Anzahl von Zugriffen stattfindet. Danache folgen noch einige weitere Zugriffe, was das Nachladen von Elementen oder Laden von weiteren Seiten mit wenigeren Elementen darstellen könnte. Ein Suchmaschienenbot wird in dieser Stufe eher weniger Zugriffe haben, da er oft nur die erste Seite aufruft. Bei einem Abruf der Seite mit WGET werden auf dieser Stufe sehr viele Zugriffe erwartet, da die komplette Seite heruntergeladen wird.

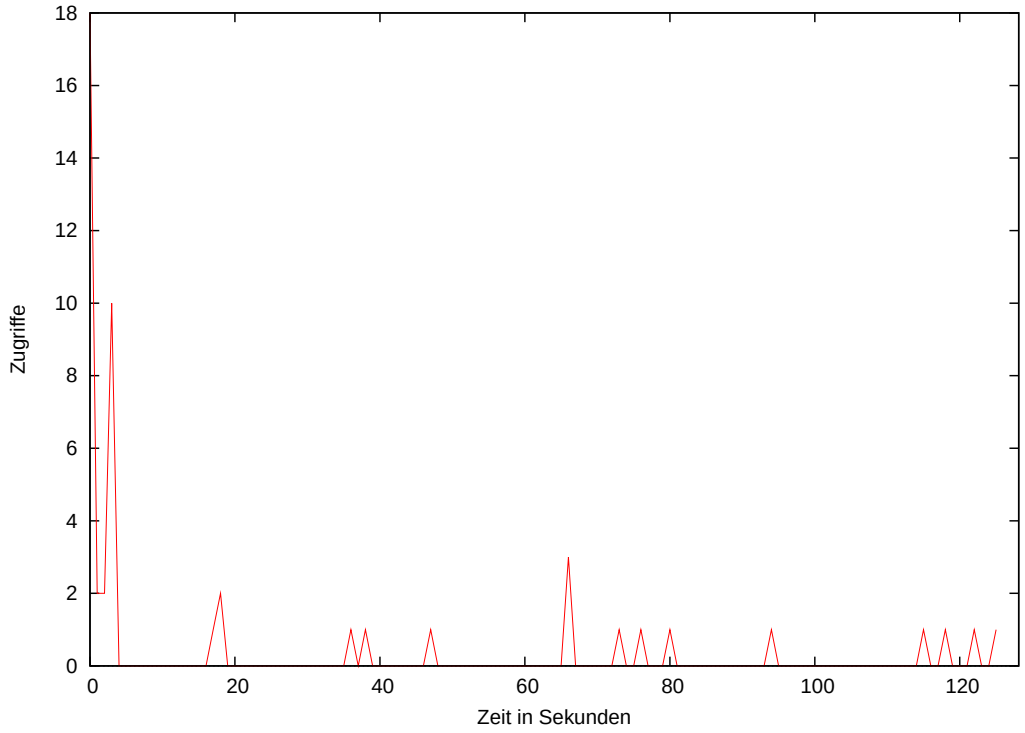

Abbildung 2.6: Zeitreihe der Zugriffe eines Benutzers auf `www.xvid.org` in der unteren Hierarchiestufe

2.3.2 Die mittlere Hierarchiestufe - der Besuch

Ein Besuch einer Webseite ist die Menge aller Zugriffe bis der Benutzer die Seite verlässt, also für einen längeren Zeitraum keinen Zugriff mehr tätigt. Auch diese Ebene wird mit einem Zeitintervall und einem Timeout definiert. Das Zeitintervall hat eine Größe von zehn Sekunden. Es wurde größer gewählt als das Intervall der ersten Stufe, um einzelne Elemente einer Seite zusammenzufasssen. Als Timeout wurden fünfzehn Minuten gewählt. Wenn von einer IP Adresse fünfzehn Minuten lang kein Zugriff ausgeführt wird, wird die Zeitreihe beendet. Ein Beispiel für eine Zeitreihe in dieser Hierarchieebene ist in Abbildung 2.7. Die Reihe ist von der selben IP Adresse wie in Abbildung 2.6. Hier erkennt man den ersten Aufruf innerhalb der ersten zwei Minunten. Dann taucht eine Pause auf, in der keine Zugriffe stattfinden. Nach dieser Pause gibt es wieder Zugriffe, bis der Besuch beendet wird.

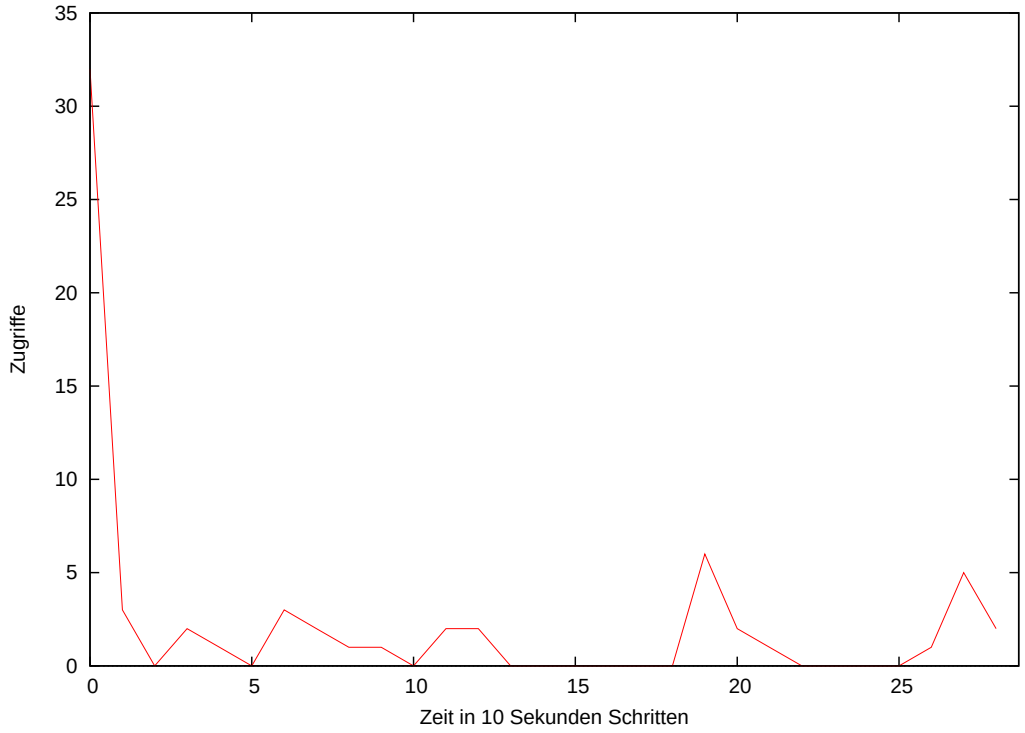

Abbildung 2.7: Zeitreihe der Zugriffe eines Benutzers auf `www.xvid.org` in der mittleren Hierarchiestufe

Allerdings können sehr kurze, aggressive Angriffe auf dieser Stufe nicht mehr gut modelliert werden. Bei einem rekursiven Laden einer komplette Seite werden möglicherweise alle Dateien schon innerhalb von zehn Sekunden geladen. Somit würde die resultierende Zeitreihe aus nur einem Wert bestehen. Ein ARIMA Modell kann damit nicht erzeugt werden.

2.3.3 Die obere Hierarchiestufe - wiederkehrende Besuche

Auf der letzten Stufe sollen mehrere, wiederkehrende Besuche modelliert werden. Hiermit ist das wiederholte Aufrufen der Seite über einen größeren Zeitraum hinweg gemeint. Dieses Verhalten wird hauptsächlich von Benutzern und gutartigen Bots erwartet. Der Unterschied könnte die Menge der Zugriffe in dem gewählten Zeitintervall sein. Von Benutzern erwartet man eventuell mehr Zugriffe als von Bots, da sich Benutzer länger auf der Seite aufhalten und sich mehr Elemente anschauen. Das gewählte Zeitintervall beträgt zwölf Stunden, da auf dieser Stufe nur das Wiederkehren in einem größen Zeitraum von Interesse ist. Ein kleineres Zeitintervall würde eventuell wieder zu vielen Nullen zwischen den Werten führen. Ein Timeout wird hier nicht gewählt, da der komplette Zeitraum betrachtet werden soll. Es wird außerdem erwartet, dass Angriffe in dieser Stufe kaum eine Rolle spielen werden, da sie nur einmal stattfinden und nicht mehrmals die Seite besuchen. Die Abbildung 2.8 zeigt wieder die Zugriffe des Benutzers auf `www.xvid.org`. Hier kann man sehr deutlich die einzelnen Besuche der IP Adresse auf die Seite erkennen.

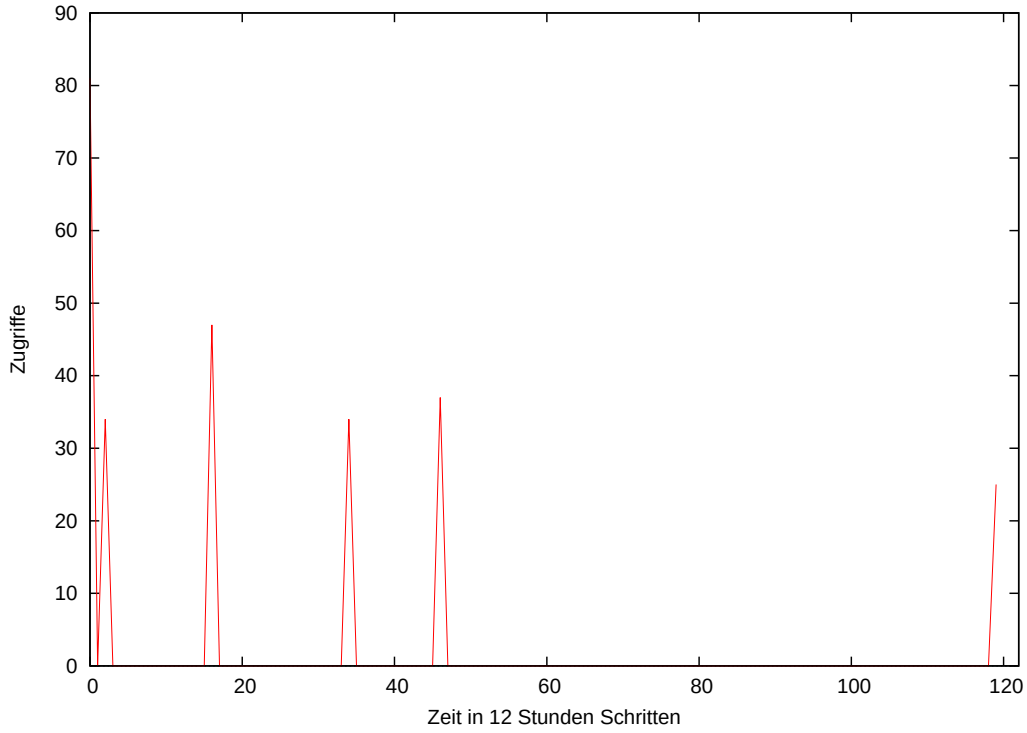

Abbildung 2.8: Zeitreihe der Zugriffe eines Benutzers auf `www.xvid.org` in der oberen Hierarchiestufe

Kapitel 3

Ergebnisse

In diesem Kapitel werden die Ergebnisse vorgestellt, die durch das Anwenden der im vorherigen Kapitel beschriebenen Modelle entstanden sind. Die Daten sind wie in Abschnitt 1.2 beschrieben von `www.xvid.org` bzw. sind im Falle der Angriffe selbst erzeugt worden. Zu jeder IP Adresse, die mehr als hundert Zugriffe in der Logdatei hat, wird für jede Ebene eine Zeitreihe erzeugt. Es werden die Zugriffe in dem jeweiligen Zeitintervall beginnend beim ersten Zugriff gezählt. Übersteigt die Differenz zum nächsten Zugriff den festgelegten Timeout, wird die Reihe abgeschlossen. Auf diese Reihe werden die im vorigen Kaptiel beschriebenen Modelle angewandt und schließlich die erzeugten Features zum Aufbau und zur Klassifikation mit einem Entscheidungsbaum eingesetzt.

3.1 Untere Hierarchiestufe

Dieser Abschnitt fasst die Ergebnisse der Modelle und der Klassifikation der Zeitreihen auf der unteren Hierarchiestufe zusammen. Im ersten Unterabschnitt werden die Ergebnissse für die Beschreibung der Zeitreihe mit Erwartungswert, Varianz und der Länge der Zeitreihe beschrieben. Darauf folgen die Ergebnisse mit den ARIMA Modellen. Im dritten Unterabschnitt werden die Ergebnisse noch kombiniert, um zu prüfen, ob sich die Ergebnisse dadurch noch verbessern lassen.

3.1.1 Erwartungswert und Varianz

Abbildung 3.1 zeigt einen Plot der Erwartungswerte und dazugehörigen Varianzen der verschiedenen Klassen. Die Länge der Zeitreihen wird in dem Plot nicht berücksichtigt. Man kann gut erkennen, dass sich die beiden verschiedenen Angriffsarten auf einige Gebiete beschränken. So sind die verteilten Angriffe sehr nahe am Ursprung des Koordinatensystems aufgetragen. Die Zugriffe erfolgen alle mit einem geringen Erwartungswert zwischen null und zwei Zugriffen pro Sekunde und einer ebenfalls geringen Abweichung davon davon. Einen großen Teil der WGET Angriffe haben einen Erwartungswert von 13 und 14 Zugriffen pro Sekunde. Die Abweichung von diesem Wert ist höher als bei den verteilten Attacken, was an den beiden "Balken" erkennbar ist. Die restlichen Attacken zeichnen sich nicht so deutlich ab. Die Benutzer und Bots verteilen sich auf das ganze Gebiet. Sie lassen sich nicht eindeutig voneinander abgrenzen.

In den Tabellen 3.1 und 3.2 sind die Ergebnisse der Klassifikation mit einem Entscheidungsbaum mit den Merkmalen Erwartungswert, Varianz und Länge der Zeitreihe angegeben. Die ersten vier Spalten der Tabelle geben die jeweilige Konfusionsmatrix an. Die erste Zeile gibt dabei an, wieviele Benutzer als Benutzer,

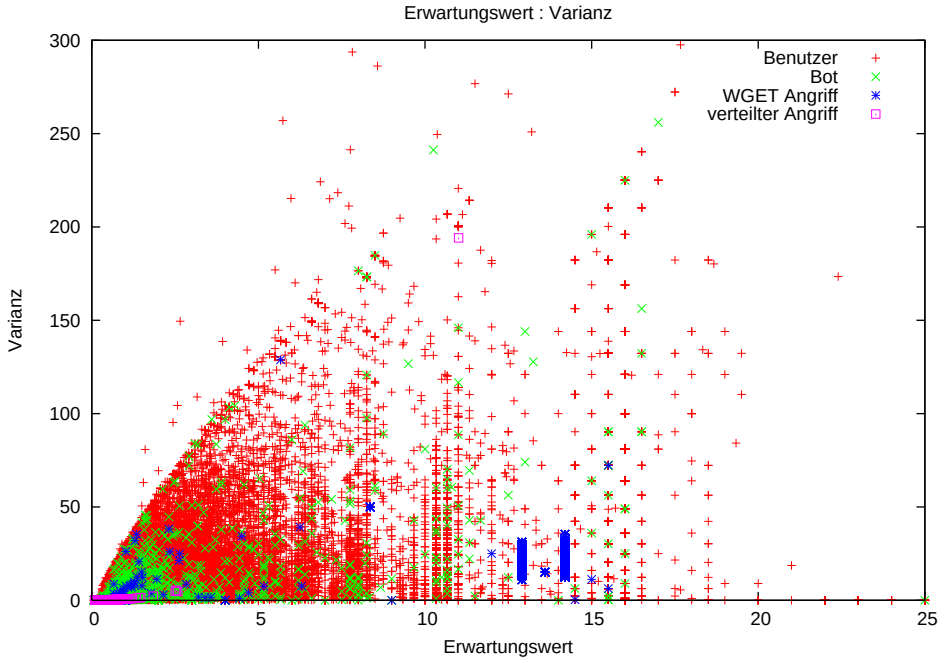

Abbildung 3.1: Erwartungswert und Varianz der unteren Ebene

WGET-Angriff, verteilter Angriff oder Bot erkannt wurden. In der zweiten Zeile folgen diese Angaben für die Angriffe mit WGET, usw. In der letzten Spalte werden die Erkennungsraten für die jeweiligen Klasse angegeben. Insgesamt ergibt sich eine Erkennungsrate von 98.11% bei den Trainingsdaten und ebenfalls eine Erkennungsrate von 98.11% bei den Testdaten. Die Ergebnisse der Trainingsdaten basieren auf zehnfacher Cross-Validation. Es ist sehr auffällig, dass die Bots nur sehr schlecht erkannt werden, und die meisten von ihnen als Benutzer klassifiziert werden. In dem Plot, der Erwartungswert und Varianz gegenüberstellt, wurde schon deutlich, dass sich die beiden Klassen, anhand der Parameter, nicht deutlich unterscheiden. Bei einem Entscheidungsbaum wird an den Blättern die Klasse gewählt, die stärker repräsentiert wird. Da in den Trainingsdaten mehr Benutzer als Bots vorkommen, werden somit an den Blättern häufiger Benutzer klassifiziert. Die Angriffe werden, ebenso wie die Benutzer, sehr gut erkannt.

Tabelle 3.1: Erkennungsraten mit Erwartungswert und Varianz der untere Ebene auf den Trainingsdaten

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	27276	6	2	1	99.97%
WGET	91	4423	1	0	97.96%
verteilt	3	1	39996	0	99.99%
Bot	1264	2	7	23	1.78%

Tabelle 3.2: Erkennungsraten mit Erwartungswert und Varianz der untere Ebene auf den Testdaten

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	6820	0	0	0	100%
WGET	26	1102	0	0	97.7%
verteilt	0	0	10000	0	100%
Bot	320	0	0	4	1.23%

3.1.2 ARIMA Modelle

Für die Klassifikation mit ARIMA Modellen werden zunächst verschiedene ARIMA-(p,d,q) Modelle für die einzelnen Zeitreihen jeder Klasse getestet. Für jedes Modell werden die Koeffizienten ϕ_i und θ_i sowie eine Konstante ϵ (das weiße Rauschen) berechnet und sowohl der AIC als auch der BIC Wert bestimmt. Nun wird für jede Klasse der Mittelwert des AIC und BIC berechnet. Das Modell mit minimalen Mittelwerten für AIC und BIC wird dann zur Klassifikation mit einem Entscheidungsbaum ausgewählt, d.h. die Koeffizienten werden als Features für den Baum verwendet. Die Idee dahinter ist, dass das Modell, welches die Zeitreihen am Besten modelliert auch das Modell ist, das die Features liefert, die am geeignetsten für die Klassifikation sind. Tabelle 3.3 listet die AIC Mittelwerte der Klassen für verschiedene Modelle auf, Tabelle 3.4 die Mittelwerte des BIC. Sowohl AIC als auch BIC bevorzugen das ARIMA(1,0,1) Modell für die Klassen Benutzer und Bot. Für die Klasse WGET-Angriff liefert das ARIMA(1,1,1) Modell den besten AIC und BIC Mittelwert, bei den verteilten Angriffen das ARIMA(2,0,2) Modell. Dabei ist allerdings zu beachten, dass für die Zeitreihen, für die das Modell nicht bestimmt werden kann, auch kein AIC/BIC berechnet werden kann.

Tabelle 3.3: AIC Mittelwerte verschiedener ARIMA Modelle für die Zugriffe der unteren Ebene

Modell	Benutzer	Bot	WGET-Angriff	verteilter Angriff
ARIMA(1,0,1)	**141.27**	**152.16**	893.38	3402.69
ARIMA(1,1,1)	144.41	159.45	**882.66**	5169.58
ARIMA(2,0,2)	156.19	163.78	889.84	**1795.69**
ARIMA(2,1,2)	161.67	182.30	759.76	3126.44

Tabelle 3.4: BIC Mittelwerte verschiedener ARIMA Modelle für die Zugriffe der unteren Ebene

Modell	Benutzer	Bot	WGET-Angriff	verteilter Angriff
ARIMA(1,0,1)	**145.6**	**156.71**	895.08	3429.74
ARIMA(1,1,1)	148.76	164.07	**883.9**	5191.19
ARIMA(2,0,2)	163.93	172.10	892.11	**1836.26**
ARIMA(2,1,2)	169.62	190.96	761.78	3166.55

Tabelle 3.5: Erkennungsraten mit ARIMA(1,0,1) der unteren Ebene auf den Trainingsdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	22169	25	37	20	81.25% (99,63%)
WGET	89	4367	0	2	96.72% (97.96%)
verteilt	42	0	39957	1	99.89%
Bot	703	4	8	7	0.54% (0.97%)

Tabelle 3.6: Erkennungsraten mit ARIMA(1,0,1) der unteren Ebene auf den Testdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	5522	8	7	10	80.97% (99.55%)
WGET	22	1090	1	1	96.63% (97.85%)
verteilt	6	0	9994	0	99.94%
Bot	171	0	3	1	0.31% (0.57%)

Insgesamt erscheint das ARIMA(1,0,1) Modell zuverlässiger, da es von zwei Klassen bevorzugt wird und deshalb werden die Koeffizienten dieses Modells als Merkmale für den Entscheidungsbaum verwendet. Die Ergebnisse des Baumes finden sich in Tabelle 3.5. Auffällig ist, dass die Erkennungsrate für Benutzer im Vergleich zu der Erkennungsrate mit Erwartungswert und Varianz als Features niedriger ist. Da einige Zeitreihen zu kurz sind und z.T. aus nur einem Element bestehen, können nicht für alle die Modellparameter berechnet werden. Dies kann passieren, wenn eine Seite geladen wird, die nur ein Element enthält und der Benutzer länger als zehn Sekunden keine andere Seite aufruft. Bei den Bots ist dies noch stärker zu beobachten, da sie häufig nur die Hauptseite aufrufen und keine weiteren Elemente der Seite. In Tabelle 3.5 sind aus diesem Grund zwei Erkennungsraten aufgelistet. Die erste ist die Erkennungsrate auf allen Datensätzen. Die zweite (in Klammern), ist die Erkennungsrate auf den Daten, für die das Modell berechnet werden konnte. Auch bei dem WGET-Angriff existieren Reihen, für die keine Koeffizienten berechnet werden konnten. Die Anzahl dieser war aber im Vergleich zu den Benutzern und Bots gering. In Tabelle 3.6 sind die Erkennungsraten des ARIMA(1,0,1) Modells auf den Testdaten angegeben. Sie unterscheiden sich kaum von der Ergebnissen auf den Trainingsdaten. Die Bots werden nicht erkannt. Die Benutzer sehr gut, falls die Modellparameter berechnet werden können, ebenso die WGET-Angriffe. Die verteilten Angriffe werden ebenfalls sehr gut erkannt.

Tabelle 3.7 zeigt die Erkennungsraten verschiedener ARIMA Modelle auf den Trainingsdaten zum Vergleich der Modelle und um zu prüfen, ob die Annahme, das Modell mit AIC und BIC zu wählen, sinnvoll war. Auch in dieser Tabelle werden zwei Erkennungsraten angegeben. Die erste wieder für die kompletten Daten, die zweite für die Daten, für welche die Modelle anwendbar waren. Insgesamt schneidet das ARIMA(1,0,1) Modell am besten ab. Bei den Angriffen mit WGET, erzielt das ARIMA(1,1,1) Modell bessere Ergebnisse. Diese Beobachtung deckt sich mit den Ergebnissen, die mit AIC und BIC ermittelt wurden. Ein Nachteil des ARIMA(1,1,1) Modells ist, dass durch die Differenz der Zeitreihenwerte für mehr Reihen kein Modell angepasst werden konnte. Für das ARIMA(1,0,1) Modell waren schon einige Reihen zu kurz. Damit sind für das höhere Modell nun noch mehr Reihen zu kurz.

Somit werden insgesamt weniger IP Zugriffe erfasst. Das ARIMA(2,0,2) Modell ist bei den Bots etwas besser, allerdings nur bei der Erkennungsrate für die Reihen, für die das Modell berechnet werden konnte. Bei der Erkennung von verteilten Angriffen ist es nicht besser, als ARIMA(1,0,1) obwohl AIC und BIC niedriger waren.

Tabelle 3.7: Erkennungsraten verschiedener ARIMA Modelle der unteren Ebene basierend auf den Trainingsdaten

Modell	insg.	Benutzer	WGET	vert. Angriff	Bot
1,0,1	**90.98%** (**98.62%**)	**81.25%** (**99.63%**)	96.72% (97.96%)	**99.89%** (99.89%)	**0.54%** (0.97%)
1,1,1	85.34% (98.25%)	76.51% (98.44%)	**97.28%** (**98.21%**)	92.76% (**99.96%**)	0.46% (0.87%)
2,0,2	86.99% (98.51%)	70.94% (99.40%)	96.92% (97.94%)	99.63% (99.65%)	**0.54%** (**1.13%**)
2,1,2	49.01% (97.98%)	64.85% (99.40%)	73.95 (97.80%)	36.95% (99.88%)	0.31% (0.74%)

3.1.3 Kombination

Die Erkennungsraten der Entscheidungsbäume basierend auf Erwartungswert und Varianz bzw. auf dem ARIMA(1,0,1) erzielten schon gute Erkennungsraten. Um zu prüfen ob sich diese Erkennungsrate noch verbessern kann, kombiniert man die beiden Modelle. Man baut also einen Baum auf, der $\hat{\mu}$, $\hat{\sigma}^2$, n, θ_1, ϕ_1 und ϵ_t als Features verwendet.

Tabelle 3.8: Erkennungsraten der Kombination von ARIMA(1,0,1) und Erwartungswert und Varianz der unteren Ebene auf den Trainingsdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	27270	11	1	3	99.95%
WGET	94	4418	3	0	97.85%
verteilt	3	1	39996	0	99.99%
Bot	1265	1	5	25	1.93%

Tabelle 3.9: Erkennungsraten der Kombination von ARIMA(1,0,1) und Erwartungswert und Varianz der unteren Ebene auf den Testdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	6814	4	0	2	99.91%
WGET	26	1102	0	0	97.70%
verteilt	0	0	10000	0	100%
Bot	320	0	0	4	1.23%

Die Erkennungsraten dieses Baumes finden sich in Tabelle 3.8. Vergleicht man diese Erkennungsraten mit denen der einzelnen Modellen, stellt man fest, dass sie etwas schlechter sind, als wenn man nur Erwartungswert und Varianz verwenden

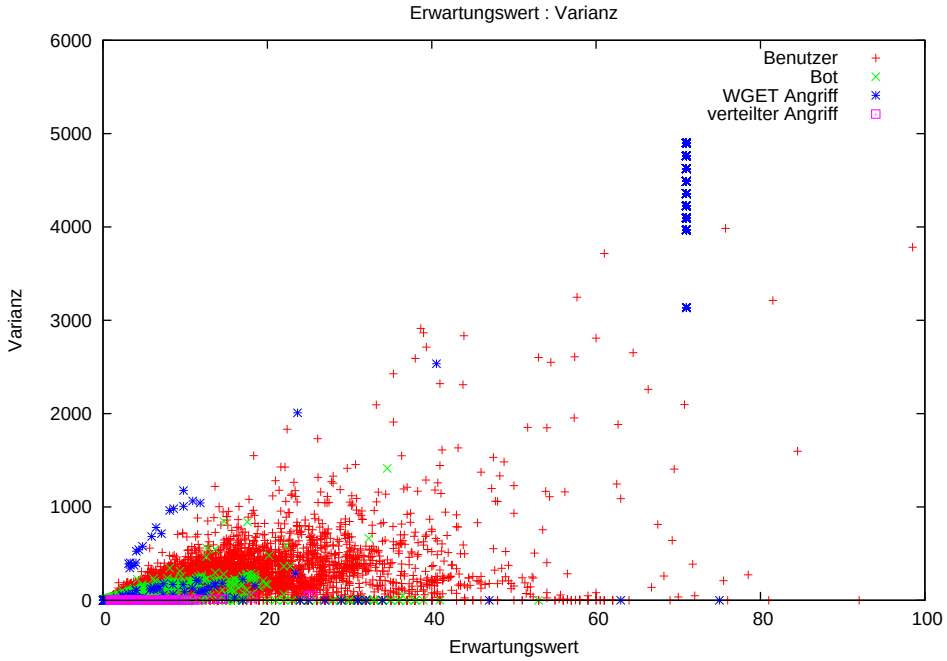

Abbildung 3.2: Erwartungswert und Varianz der mittleren Ebene

würde. Die Klassifikation ist aber besser, als nur das ARIMA Modell zu verwenden. Ein großer Vorteil ist, dass auch die Reihen zur Klassifikation verwendet werden können, deren ARIMA Modell nicht berechnet werden konnte. In diesem Fall wurde für die drei Parameter ein Wert gesetzt, der nicht durch das Modell ausgerechnet werden kann. Insgesamt ergibt sich eine Erkennungsrate von 98.1%. Die Erkennungsraten der Kombination auf den Testdaten ist in Tabelle 3.9 aufgelistet. Sie sind ebenfalls etwas schlechter als die Raten mit dem Erwartungswertmodell, aber besser als die Raten mit dem ARIMA Modell. Die gesamte Erkennungsrate beträgt 98.07%.

3.2 Mittlere Hierarchiestufe

In diesem Abschnitt werden die Ergebnisse der mittleren Ebene, also der Ebene, die einen Besuch auf die Seite modellieren soll, dargestellt. Der Abschnitt gliedert sich wieder in drei Unterabschnitte. Im ersten Abschnitt wird das Modell aus Erwartungswert und Varianz vorgestellt. Im darauf folgenden Abschnitt werden die ARIMA Modelle vorgestellt und schließlich im letzten Abschnitt die Kombination der beiden Modelle.

3.2.1 Erwartungswert und Varianz

Abbildung 3.2 stellt den Erwartungswert samt zugehöriger Varianz der vier Klassen dar. Der Plot ist dem der unteren Ebene sehr änlich. Ein Teil der WGET-Angriffe grenzt sich stark von den anderen Klassen ab. Die verteilten Angriffe sind nahe dem Ursprung und haben keine hohe Varianz. Die Benutzer und Bots sind über ein weiteres Gebiet verteilt, allerdings grenzen sie sich nicht von einander ab. Der Plot zeichnet die Punkte der Bots über die der Benutzer, weswegen man denken könnte, sie würden sich abgrenzen. Dies ist aber nicht der Fall.

Die Ergebnisse der Klassifikation mit einem Entscheidungsbaum finden sich in 3.10 und 3.11. Insgesamt ergibt sich bei dieser Hierarchiestufe eine Erkennungsrate von 98.14% auf den Trainingsdaten und eine Erkennungsrate von 98.13% auf den Testdaten. Die Ergebnisse unterscheiden sich nicht sonderlich stark von den Ergebnissen mit den Zeitreihen auf der unteren Ebene. Die Bots werden ebenfalls nicht gut erkannt.

Tabelle 3.10: Erkennungsraten mit Erwartungswert und Varianz der mittlere Ebene auf den Trainingsdaten

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	27266	6	1	12	99.93%
WGET	53	4461	1	0	98.8%
verteilt	16	0	39984	0	99.96%
Bot	1265	4	2	25	1.93%

Tabelle 3.11: Erkennungsraten mit Erwartungswert und Varianz der mittlere Ebene auf den Testdaten

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	6819	0	0	1	99.96%
WGET	16	1112	0	0	98.58%
verteilt	2	0	9997	1	99.97%
Bot	321	0	0	3	0.93%

3.2.2 ARIMA Modelle

Auch auf dieser Ebene werden zunächst die Parameter mehrerer ARIMA Modelle berechnet und dann anhand der Mittelwerte des AIC (Tabelle 3.12) und BIC (Tabelle 3.13) verglichen. Die Kriterien bevorzugen beide das ARIMA(1,0,1) Modell für die Bots und WGET-Angriffen, das ARIMA(1,1,1) Modell für die Benutzer und das ARIMA(2,0,2) Modell für die verteilten Angriffe. Auffallend sind die extrem großen Werte für WGET-Angriffe, was ein Anzeichen dafür ist, dass diese Angriffe mit der Ebene nicht sonderlich gut beschreibbar sind. Allerdings ist es nun problematisch, ein ARIMA Modell zu wählen, da keines der Modelle von mehreren Klassen bevorzugt wird. Deswegen finden sich, zum weiteren Vergleich, in Tabelle 3.14 die Erkennungsraten der Modelle auf den Trainingsdaten.

Tabelle 3.12: AIC Mittelwerte verschiedener ARIMA Modelle für die Zugriffe der mittleren Ebene

Modell	Benutzer	Bot	WGET-Angriff	verteilter Angriff
ARIMA(1,0,1)	208.53	**355.06**	**16153.39**	614.34
ARIMA(1,1,1)	**206.28**	401.15	16435.58	681.90
ARIMA(2,0,2)	237.31	370.26	16525.25	**600.19**
ARIMA(2,1,2)	238.86	370.8	18879.38	613.52
ARIMA(2,2,2)	246.83	496.57	20353.47	647.83
ARIMA(3,1,3)	272.76	463.75	20662.92	602.18

Tabelle 3.13: BIC Mittelwerte verschiedener ARIMA Modelle für die Zugriffe der mittleren Ebene

Modell	Benutzer	Bot	WGET-Angriff	verteilter Angriff
ARIMA(1,0,1)	212.40	**359.31**	**16173.23**	632.18
ARIMA(1,1,1)	**209.92**	405.21	16455.49	699.39
ARIMA(2,0,2)	244.51	378.72	16555.17	**626.69**
ARIMA(2,1,2)	246.06	378.65	18910.98	639.87
ARIMA(2,2,2)	254.09	505.06	20388.89	674.43
ARIMA(3,1,3)	284.08	476.98	20707.34	636.99

Tabelle 3.14: Erkennungsraten verschiedener ARIMA Modelle der mittleren Ebene basierend auf den Trainingsdaten

Modell	insg.	Benutzer	WGET-Angriff	verteilter Angriff	Bot
1,0,1	79.61% (97.56%)	**67.99%** (98.08%)	0.04% (1.71%)	99.08% (99.08%)	**0.46%** (**0.98%**)
1,1,1	**79.69%** (98.64%)	67.80% (99.53%)	**1.26%** (49.18%)	99.24% (**99.88%**)	0.08% (0.16%)
2,0,2	74.16% (97.2%)	55.98% (96.81%)	0% (0%)	97.33% (98.81%)	0.15% (0.40%)
2,1,2	74.64% (98.65%)	54.75% (99.21%)	0.78% (36.84%)	98.97 % (99.77%)	0.08% (0.21%)
2,2,2	73.62% (**98.94%**)	51.01% (**99.64%**)	1.57% (**75.53%**)	**99.56%** (99.86%)	0.23% (0.66%)
3,1,3	69.74% (98.82%)	43.93% (99.14%)	0.84% (44.19%)	97.38% (99.73%)	0.08% (0.28%)

Insgesamt erzielt der Entscheidungsbaum, der auf Ergebnissen des ARIMA(1,1,1) Modells aufgebaut wurde, die besten Erkennungsraten. ARIMA(2,1,2), ARIMA(2,2,2) und ARIMA(3,1,3) erzielen leicht höhere Erkennungsraten auf den Datensätzen, auf denen die Modelle angepasst werden konnten. Diese unterscheiden sich aber nur minimal von denen des ARIMA(1,1,1). Weiter fallen die sehr niedrigen Erkennungsraten der Bots auf. Diese werden, wie schon auf der unteren Ebene, weitestgehend als Benutzer klassifiziert. Die Erkennungsrate für WGET-Angriffe sind auch niedrig. Das Problem bei den Zeitreihen dieser Klasse ist, dass nur sehr wenige für ARIMA Modelle anpassbar sind, da ein Großteil der Reihen eine Länge von eins besitzt. Die Erkennungsrate der WGET-Angriffe, für die das Modell angepasst werden konnte, schwanken stark zwischen einem und 75 Prozent. Für das ARIMA(2,0,2) Modell ist die Klassifikationsrate dieser Klasse sogar Null. Die verteilten Angriffe erzielen mit allen Modellen gute Erkennungsraten. Bei den Benutzern sind die Ergebnisse der Datensätze, für die das jeweilige Modell anwendbar war, ebenfalls alle sehr gut. Allerdings können für die höheren Modelle weniger der Zeitreihen angepasst werden. Aus diesem Grund scheint das ARIMA(1,1,1) am sinnvollsten zu sein.

Tabelle 3.15: Erkennungsraten mit ARIMA(1,1,1) der mittleren Ebene auf den Trainingsdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	18498	14	69	5	67.80% (99.53%)
WGET	53	57	5	1	1.26% (49.18%)
verteilt	44	4	39695	0	99.24% (99.88%)
Bot	585	3	19	1	0.08% (0.16%)

Tabelle 3.16: Erkennungsraten mit ARIMA(1,1,1) der mittleren Ebene auf den Testdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	4620	2	17	1	67.74% (99.57%)
WGET	15	9	3	0	0.2% (33.34%)
verteilt	7	1	9920	0	99.2% (99.92%)
Bot	148	0	7	0	0% (0%)

In Tabelle 3.15 ist die Konfusionsmatrix mit den Erkennungsraten des ARIMA-(1,1,1) Modells für die einzelnen Klassen dargestellt. Die Bots werden, wie schon auf der unteren Stufe, fast alle als Benutzer erkannt. Die WGET-Angriffe werden auch zu einem großen Teil als Benutzer identifiziert. Wenn man berücksichtigt, dass nur für einen sehr kleinen Teil dieser Angriffe das Modell angewandt werden konnte, fällt dies nicht so stark ins Gewicht. Es werden nur 1.17 Prozent aller WGET-Angriffe als Benutzer erkannt. Tabelle 3.16 führt die Konfusionsmatrix und die Erkennungsraten auf den Testdaten auf. Sie sind den Raten der Trainingsdaten sehr ähnlich, nur die Bots werden gar nicht mehr erkannt. Insgesamt ergibt sich eine Erkennungsrate von 98.64 Prozent auf den Testdaten.

3.2.3 Kombination

In diesem Abschnitt werden Erwartungswert, Varianz und Länge der Zeitreihe mit den Modellparametern des ARIMA(1,1,1) Modells kombiniert. Die Ergebnisse auf den Trainingsdaten sind in Tabelle 3.17, die auf den Testdaten in Tabelle 3.18 aufgezeigt. Sie sind, wie bei der unteren Stufe, etwas schlechter, als die bei Verwendung von Erwartungswert und Varianz alleine. Die Kombination ist aber besser, als die Erkennungsraten wenn man nur das ARIMA Modell verwendet. Insgesamt sind die Erkennungsraten etwas besser, als auf der unteren Hierarchieebene. Die Erkennungsrate auf den Trainingsdaten beträgt 98.13 Prozent, auf den Testdaten 98.11 Prozent.

Tabelle 3.17: Erkennungsraten der Kombination von ARIMA(1,1,1) und Erwartungswert und Varianz der mittleren Ebene auf den Trainingsdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	27264	7	1	13	99.92%
WGET	59	4455	1	0	98.67%
verteilt	16	0	39984	0	99.96%
Bot	1264	4	2	26	2.01%

Tabelle 3.18: Erkennungsraten der Kombination von ARIMA(1,1,1) und Erwartungswert und Varianz der mittleren Ebene auf den Testdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	6818	0	0	2	99.97%
WGET	19	1109	0	0	98.32%
verteilt	2	0	9997	1	99.97%
Bot	321	0	0	3	0.94%

3.3 Obere Hierarchiestufe

Dieser Abschnitt stellt die Ergebnisse, die mit der oberen Hierarchiestufe erzielt wurden, vor.

3.3.1 Erwartungswert und Varianz

Die Tabellen 3.19 und 3.20 zeigen die Ergebnisse der Klassifikation mit einem Entscheidungsbaum. Die Erkennungsrate auf den Trainingsdaten beträgt 98.68%. Im Vergleich zu der Erkennungsrate der Erwartungswerte und Varianzen mit den niedrigeren Ebenen, fällt auf, dass die verteilten Angriffe alle richtig erkannt werden. Dies resultiert daraus, dass die Zugriffszeiten normalverteilt generiert wurden mit einer festen Länge. Alle diese Zugriffe fallen in das erste Intervall, da der Angriff keine zwölf Stunden lang andauert. Da anschließend auch keine weiteren Zugriffe erfolgen, besteht die Zeitreihe nur aus einem Wert zu einem Zeitintervall. Der Mittelwert ist also für alle IP Adressen dieser Klasse gleich und somit kann der Entscheidungsbaum diese Adressen sehr gut von den anderen unterscheiden. Dies ist zwar ein gutes Ergebnis, in der Praxis aber nicht einsetzbar. Die real ausgeführten Angriffe werden nicht die gleiche Anzahl von Zugriffen besitzen. Außerdem ist es nicht das Ziel der oberen Ebene, die Angriffe zu erkennen, sondern das wiederkehrende Besuchen der Seite von Benutzern und Bots. Zur Vollständigkeit werden die Ergebnisse, die Angriffe betreffen, sofern vorhanden, trotzdem aufgeführt.

Weiter ist auffallend, dass die Bots mit dem Erwartungswert und der Varianz auf dieser Ebene besser erkannt werden, als auf den beiden anderen Stufen. So werden Bots mit circa 40% richtig klassifiziert. Die Erkennungsrate auf den kompletten Testdaten beträgt 98.69%. Die Klassen Benutzer und WGET-Angriff werden insgesamt ein klein wenig schlechter erkannt im Vergleich zu den anderen Ebenen.

Tabelle 3.19: Erkennungsraten mit Erwartungswert und Varianz der obere Ebene auf den Trainingsdaten

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	27156	45	0	84	99.527%
WGET	76	4435	1	4	98.228%
verteilt	0	0	40000	0	100%
Bot	740	6	1	539	41.5%

Tabelle 3.20: Erkennungsraten mit Erwartungswert und Varianz der obere Ebene auf den Testdaten

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	6797	10	0	13	99.663%
WGET	21	1105	2	0	97.961%
verteilt	0	0	10000	0	100%
Bot	192	1	0	131	40.432%

3.3.2 ARIMA

Die Tabelle 3.21 mit den AIC Mittelwerten zeigt, dass unterschiedliche Modelle bei verschiedenen Klassen bevorzugt werden. So erzielt das ARIMA(2,1,2) Modell den niedrigsten Mittelwert für die Benutzer, während das ARIMA(2,0,2) Modell bei den Bots den besseren Wert erreicht. Überraschend sind die im Vergleich zu den anderen Ebenen niedrigen AIC und BIC Werte für die WGET-Angriffe. Es ist auch interessant, dass ARIMA Modelle für die Zeitreihen der Angriffe berechnet werden konnten. Diese Angriffe haben zwar keine feste Anzahl der Zugriffe, wie die verteilten Angriffe, aber sie sind trotzdem kürzer als zwölf Stunden. Der Grund dafür, dass trotzdem ein Modell aufgestellt werden kann ist, dass manche IP Adressen mehrfach für einen Angriff verwendet wurden und dies in der Modellierung auf der oberen Hierarchiestufe auftaucht. Für die verteilten Angriffe konnten keine Modellparameter berechnet werden. Sie fallen somit komplett bei der Klassifikation weg. Das BIC ist ebenfalls für Benutzer minimal bei dem ARIMA(2,1,2) Modell und für die Bots minimal bei ARIMA(2,0,2) (Tabelle 3.22). Es ist also kein ARIMA Modelle eindeutig zu bevorzugen. Deswegen werden wieder die Erkennungsraten auf den Trainingsdaten verglichen. Die Ergebnisse dazu finden sich in Tabelle 3.23.

Tabelle 3.21: AIC Mittelwerte verschiedener ARIMA Modelle der Zugriffe auf der oberen Ebene

Modell	Benutzer	Bot	WGET-Angriff	verteilter Angriff
ARIMA(1,0,1)	655.10	962.44	55.60	-
ARIMA(1,1,1)	657.78	992.82	**41.58**	-
ARIMA(2,0,2)	652.22	**900.2**	112.38	-
ARIMA(2,1,2)	**643.29**	926.15	98.72	-
ARIMA(2,2,2)	671.53	964.86	93.84	-
ARIMA(3,1,3)	712.49	933.02	106.86	-

Tabelle 3.22: BIC Mittelwerte verschiedener ARIMA Modelle der Zugriffe auf der oberen Ebene

Modell	Benutzer	Bot	WGET-Angriff	verteilter Angriff
ARIMA(1,0,1)	663.97	973.20	43.17	-
ARIMA(1,1,1)	666.56	1004.21	**38.03**	-
ARIMA(2,0,2)	665.58	**915.25**	114.09	-
ARIMA(2,1,2)	**656.36**	942.24	98.61	-
ARIMA(2,2,2)	685.06	981.35	94.12	-
ARIMA(3,1,3)	731.38	954.64	108.55	-

Tabelle 3.23: Erkennungsraten verschiedener ARIMA Modelle der oberen Ebene basierend auf den Trainingsdaten

Modell	insg.	Benutzer	WGET	Bot
1,0,1	58.43% (**94.29%**)	**64.38%** (99.14%)	35.73% (96.53%)	12.35% (14.27%)
1,1,1	**58.59%** (94.13%)	64.23% (99.90%)	**41.22%** (**97.49%**)	0.39% (0.44%)
2,0,2	49.16% (93.7%)	59.1% (99.11%)	0.09% (12.5%)	10.8% (13.21%)
2,1,2	47.21% (93.47%)	57.11% (99.77%)	0% (0%)	3.24% (3.93%)
2,2,2	48.78% (93.92%)	58.53% (99.45%)	0.13% (21.43%)	**13.04%** (**15.32%**)
3,1,3	38.35% (92.45%)	46.50% (**99.91%**)	0.02% (5.88%)	0.30% (0.40%)

Insgesamt schneidet das ARIMA(1,1,1) Modell am besten ab. Mit diesem Modell werden aber gerade die Bots wieder sehr schlecht erkannt. Die besten Erkennungsraten für die Bots erzielt das ARIMA(2,2,2) Modell. Der Nachteil bei diesem Modell ist, dass wieder weniger Zeitreihen betrachtet werden können, als z.B. mit dem ARIMA(1,0,1) Modell. Insgesamt sind die Erkennungsraten dieses Modells höher, weswegen wir es bevorzugen. Die genauen Ergebnisse der Klassifikation mit ARIMA(1,0,1) Parametern findet sich für die Trainingsdaten sind in Tabelle 3.24 und für die Testdaten in Tabelle 3.25. Man erhält mit diesem Modell eine Erkennungsrate von 94.39 Prozent auf den Testdaten, wenn man alle Datensätze betrachtet reduziert sich die Rate auf 58.99 Prozent.

Tabelle 3.24: Erkennungsraten mit ARIMA(1,0,1) der oberen Ebene auf den Trainingsdaten

	Benutzer	WGET-Angriff	Bot	Erkennungsrate
Benutzer	17566	56	97	64.38% (99.14%)
WGET-Angriff	58	1613	0	35.73% (96.53%)
Bots	957	4	160	12.35% (14.27%)

Tabelle 3.25: Erkennungsraten mit ARIMA(1,0,1) der oberen Ebene auf den Testdaten

	Benutzer	WGET-Angriff	Bot	Erkennungsrate
Benutzer	4441	12	35	65.12% (98.95%)
WGET-Angriff	15	380	0	33.69% (96.2%)
Bots	957	4	160	18.21% (20.56%)

3.3.3 Kombination

Wie auch bei den beiden anderen Hierarchiestufen werden die beiden bisherigen Modelle zusammengefasst. Die Ergebnisse davon sind in den Tabellen 3.26 und 3.27. Die hundertprozentige Erkennungsrate der verteilten Angriffe erklärt sich wieder durch die gleiche Länge aller dieser Angriffe, ist also nicht Praxis relevant. Auffallend ist die verhältnismäsig hohe Erkennungsrate für die Bots im Vergleich zu den anderen Ebenen.

Tabelle 3.26: Erkennungsraten der Kombination von ARIMA(1,0,1) und Erwartungswert und Varianz der obere Ebene auf den Trainingsdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	27152	58	0	75	99.51%
WGET	59	4451	0	5	98.58%
verteilt	0	0	40000	0	100%
Bots	767	5	1	523	40.35%

Tabelle 3.27: Erkennungsraten der Kombination von ARIMA(1,0,1) und Erwartungswert und Varianz der oberen Ebene auf den Testdaten)

	Benutzer	WGET	verteilt	Bot	Erkennungsrate
Benutzer	6739	16	0	11	98.81%
WGET	191	1111	2	14	98.49%
verteilt	0	0	10000	0	100%
Bots	191	1	0	132	40.74%

Kapitel 4

Zusammenfassung und Ausblick

In diesem letzten Kapitel werden zunächst noch einmal die Ergebnisse der Arbeit im ersten Abschnitt zusammengefasst. Der zweite Abschnitt gibt einen Ausblick über mögliche Weiterführungen dieser Arbeit.

4.1 Zusammenfassung der Ergebnisse

Die vorigen Kapitel haben gezeigt, dass man die HTTP Zugriffszeiten als Zeitreihe in drei Hierarchiestufen modellieren kann. Auf den resultierenden Zeitreihen kann man Modelle zur Zeitreihenanalyse anwenden und deren Ergebnisse können zur Klassifikation mit einem Entscheidungsbaum herangezogen werden. So werden insgesamt Klassifikationsraten von 98 Prozent bei den Zugriffen auf `www.xvid.org` erreicht.

Von den verwendeten Modellen erwies sich das Modell aus Erwartungswert, Varianz und Länge der Zeitreihe am effizientesten. Dieses Modell erzielte die besten Erkennungsraten auf allen Ebenen. Ein großer Vorteil dieses Modells ist, dass es sehr einfach und ohne großen Aufwand berechenbar ist. Ein weiterer Vorteil ist, dass man dieses Modell auf jede Zeitreihe anwenden kann, im Gegensatz zu den ARIMA Modellen, die für einige Zeitreihen nicht berechenbar waren.

Das ARIMA Modell erzielte auch gute Ergebnisse bei der anschließenden Klassifikation. Im Vergleich zu dem Modell aus Erwartungswert und Varianz hat es aber einige Nachteile. Ein Nachteil ist, ein geeignetes Modell zu finden, dass möglichst viele der Zeitreihen gut beschreibt. Die AIC und BIC Werte haben gezeigt, dass die verschiedenen Klassen in einer Ebene durch verschiedene ARIMA Modelle besser beschrieben werden. So ist nach den AIC und BIC Mittelwerten das ARIMA(1,0,1) Modell am Besten für Benutzer und Bots. Für die verteilten Angriffe wird ARIMA(2,0,2) bevorzugt und für die WGET Angriffe ARIMA(1,1,1). Bei den anderen beiden Ebenen ist dies ähnlich. Auch ist es schwierig ein Modell zu finden, dass alle Reihen einer Klasse gut beschreibt. Um die Parameter für die Klassifikation zu berechnen, wurde für jede Ebene ein einzelnes Modell gewählt, um dadurch eine bessere Vergleichbarkeit der Werte zu gewährleisten. Die Entscheidungsbäume erzielen aber trotz dieser Nachteile gute Erkennungsraten.

Die Kombination der beiden Modelle erzielte keine Verbesserung im direkten Vergleich mit dem einfachen Modell des Erwartungswertes und der Varianz. Allerdings war die Kombination nicht schlechter als das ARIMA Modell allein, sondern lag recht nahe an den Erkennungsraten des ersten Modells.

Die Modellierung der Zugriffszeiten in drei Hierarchiestufen erwies sich als gute Wahl. In der unteren Ebene werden die Angriffe am Besten erkannt. Die Zugriffsraten der Angriffe unterscheiden sich sehr stark von denen der Benutzer. Die beiden Angriffsarten lassen sich auch sehr gut voneinander unterscheiden. Allerdings lassen sich die Bots auf dieser Ebene nicht erkennen. Dies resultiert aus dem Grund, dass die Zugriffsraten von Benutzern und Bots sehr ähnlich sind. Es gibt Benutzer, die sich nur die Hauptseite anschauen und dann keine weiteren Zugriffe mehr tätigen. Dann gibt es keinen großen Unterschied zu einem Bot, der auch alle Elemente der Hauptseite aufruft. Außerdem gibt es viel weniger Bots in den Trainingsdaten als Benutzer, somit werden viele Bots den Benutzern zugeordnet.

Die mittlere Ebene kann nicht mehr zur Klassifikation der WGET Angriffe verwendet werden, allerdings lassen sich die verteilten Angriffe und die Benutzer sehr gut erkennen. Bots lassen sich in dieser Ebene ebenfalls nicht erkennen aus den selben Gründen wie bei der unteren Ebene. Sie sind dafür in der oberen Ebene besser erkennbar, in welcher die Angriffe nicht mehr modellierbar sind. Die Angriffe sind für diese Ebene zu kurz, da ein Angriff in der Regel nicht über mehrere Tage hinweg andauert. Bots haben über einen längeren Zeitraum hinweg ein anderes Zugriffsverhalten als Benutzer. So dürften sie sich in der Regelmäsigkeit als auch der Anzahl der Zugriffe von den menschlichen Zugriffen unterscheiden. Diese Unterschiede werden in der oberen Ebene deutlich, was in einer höheren Erkennungsrate im Vergleich zu den beiden anderen Ebenen resultiert.

Was man bei den Ergebnissen noch beachten sollte ist, dass die Seite `www.xvid.org` einen Videocodec vorstellt. Es ist also keine Webseite, von der man hohe, stark regelmäßige Besuche von menschlichen Benutzern erwartet. Auf einer Webseite, die Nachrichten verbreitet, würde man ein regelmäßigeres Besucherverhalten erwarten. Die Zugriffe auf eine solche Seite lassen sich eventuell besser als Zeitreihe interpretieren, auf die ARIMA Modelle angewandt werden können. Gerade dürften die Zeitreihen in der oberen Hierarchiestufe viel aussagekräftiger werden, als die, die bei dieser Arbeit verwendet wurden.

Zusammenfassend kann man sagen, dass die Ebenen das unterscheiden, was sie unterscheiden sollen. Die untere Ebene ist für die sofortigen Zugriffe, also unterscheiden sich hier die Angriffe von den Benutzern. Die mittlere Ebene ist für einen Besuch, unterscheidet also die verteilten Angriffe von den Benutzern. In der oberen Ebene lassen sich schließlich die Bots von den Benutzern unterscheiden.

4.2 Ausblick auf mögliche Weiterführungen der Arbeit

Der vorige Abschnitt hat noch einige Ungereimtheiten offen gelegt. Dieser Abschnitt geht auf diese ein und stellt möglich Lösungsansätze vor. Desweiteren wird noch auf die praktische Anwendung des Ansatzes kurz eingegangen.

Am auffälligsten an den Ergebnissen ist, dass sich die Bots kaum von den Benutzern unterscheiden lassen. Ein Erklärung dafür ist, dass die Zugriffsraten der Bots denen der Benutzer ähnlich sind und folglich die Unterscheidung sehr schwierig ist. Ein weiterer Grund ist die geringe Repräsentierung in den Trainingsdaten im Vergleich zu den Benutzern. Eine Möglichkeit die Erkennungsrate zu verbessern, wäre eine gleiche Anzahl von Trainingsdaten der beiden Klassen zu verwenden, indem man z.B. weniger Benutzerdatensätze oder mehr Datensätze für Bots verwendet. Dies könnte man umsetzen, indem man die bestehenden Zeiten kopiert und minimal abändert, um so generierte Botzugriffe zu erhalten. Allerdings wird dann nicht mehr das in der Realität existierende Verhältnis von Benutzern zu Bots berücksichtigt.

Eine weitere Möglichkeit wäre den Entscheidungsbaum so zu modifizieren, dass er bevorzugt Bots vor Benutzern erkennt, indem man z.B. Gewichte einführt, die die Entscheidung für Bots erhöht. Dabei sollte aber berücksichtigt werden, dass das Hauptziel ist, die Benutzer von den Angriffen zu unterscheiden, um so die Sicherheit der Webseiten zu erhöhen. Die Klasse der Bots umfasst gutartige automatisierte Zugriffe auf die Seite, die keinen Schaden anrichten. Im Anwendungsfall ist es das Ziel die Angriffe früh zu erkennen und deren Zugriffe zu stoppen. Dabei sollten allerdings möglichst keine Benutzer falsch erkannt werden, da sie weiter auf die Seite zugreifen sollen. Bots würde man vor den Benutzern sperren um deren Zugriffe zu unterbinden, aber diese Zugriffe werden in der Regel nicht sonderliche viele Ressourcen in Anspruch nehmen. Es ist also wichtiger die Benutzer richtig zu erkennen, als die Bots.

Ein weiteres auffallendes Ergebnis ist, dass es kein ARIMA Modell gibt, das für alle Klassen gleich gut ist. Eine Möglichkeit dies zu beheben, wäre des Berechnen der Parameter verschiedener ARIMA Modelle. Man könnte dann für jedes Modell einen eigenen Entscheidungsbäume verwenden und die Klasse zuordnen, die von den meisten Bäumen gewählt wird. Eine weitere Möglichkeit ist auch die Wahl einer anderen Methode zur Zeitreihenanalyse.

Eine weitere Frage ist die Praxisrelevanz des vorgestellten Ansatzes. Ein genereller Nachteil der Klassifikation von HTTP Zugriffen ist, dass sie nicht während des Aufrufs auf eine Webseite stattfindet, sondern erst später. Um einen Angriff zu erkennen und eine eventuelle Gegenmaßnahme einleiten zu können, ist es aber wichtig, dass der Angriff in Echtzeit erkannt werden kann. Dazu müssten die Modelle auf die TCP Pakete angewandt werden. Anstelle der HTTP Zugriffszeiten wären die IP-Paketzeiten die Basis für den Modellaufbau.

Für die Klassifikation von Angriffen bietet sich die untere Ebene am ehesten an. Die Intervallgröße in dieser Ebene beträgt eine Sekunde. Eine mögliche Frage wäre, ob man dies einfach so übernehmen kann, oder ob es sinnvoll wäre, eine höhere Intervallgröße zu verwenden. Auch stellt sich die Frage, wie viele Pakete in die Analyse mit einbezogen werden müssen, um eine verlässliche Aussage zu bekommen. Interessant wäre auch, herauszufinden, ob man den Klassifikator auf den HTTP Zugriffszeiten trainieren und ihn für die Klassifikation der IP-Paketzeiten verwenden kann. Natürlich stellt sich auch die Frage, ob diese Idee generell umsetzbar ist und welches Zeitreihenanalysemodell und welchen Klassifikator man am sinnvollsten dafür verwendet.

Insgesamt wurde in dieser Arbeit gezeigt, dass die HTTP Zugriffszeiten mit Zeitreihen modelliert werden können und dass es sich anbietet, verschiedene Hierarchien zu verwenden, die verschiedene Stufen der Zugriffe widergeben.

Literaturverzeichnis

[1] Sabyasachi Basu, Amarnath Mukherjee, and Steve Klivansky. Time series models for internet traffic. In *INFOCOM (2)*, pages 611–620, 1996.

[2] J. Beran, R.J. Bhansali, and D. Ocker. On unified model selection for stationary and nonstationary short- and long-memory autoregressive processes . In *Biometrika*, volume 85, pages 921–934, April 1998.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley-Interscience, 2001.

[4] Network Working Group. http://www.faqs.org/rfcs/rfc1413.html, 1993.

[5] Network Working Group. http://tools.ietf.org/html/rfc1945, 1996.

[6] Network Working Group. http://tools.ietf.org/html/rfc2616, 1999.

[7] Cheng Guang, Gong Jian, and Ding Wei. A Time-Series Decomposed Model of Network Traffic. In *ICNC*, pages 338–345, 2005.

[8] Ulrich Helfenstein. Box-Jenkins Modelling of some viral infectious diseases. In *Statistics in medicine*, volume 5, pages 37–47, 1986.

[9] Martijn Koster. http://www.robotstxt.org/orig.html.

[10] Danny McPherson, Dr. Craig Labovitz, and Mike Hollyman. Worldwide infrastructure security report, September 2007.

[11] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. In *ACM SIGCOMM Computer Communications Review*, volume 34, pages 39–54, April 2004.

[12] Bernhard Pfaff. *Analysis of integrated and cointegrated time series with R.* Springer, 2006.

[13] Apache HTTP Server Project. http://httpd.apache.org/docs/1.3/logs.html.

[14] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and its applications - Second Edition.* Springer, 2006.

[15] Andrew S. Tanenbaum. *Computer Networks - Fourth Edition.* Pearson Education International, 2003.

[16] Hao Yin, Chuang Lin, Berton Sebastien, Bo Li, and Geyong Min. Network traffic prediction based on a new time series model. In *Inernational Journal of Communication Sytems*, volume 18, pages 711–729, 2005.