# Systemadministration: IDS-Pi

Benedikt Geiger, Ruben Miller

Dezember 2023


Abbildung 1: IDS-Pi

# Inhaltsverzeichnis


**1 Einleitung** **3**
- 1.1 Motivation . . . 3
- 1.2 Code Repository . . . 3

**2 Anforderungen** **3**
- 2.1 Unabdingliche Anforderungen . . . 3
  - 2.1.1 File Based Intrusion Detection . . . 3
  - 2.1.2 Network Intrusion Detection . . . 3
  - 2.1.3 Benachrichtigungen . . . 3
- 2.2 Optionale Anforderungen . . . 4
  - 2.2.1 Oberflächen . . . 4

**3 Verwendete Technologien** **5**
- 3.1 SSH . . . 5
- 3.2 Raspberry Pi . . . 5
- 3.3 File Based Intrusion Detection: AIDE . . . 5
- 3.4 Network Intrusion Detection: Suricata . . . 7
- 3.5 E-Mail . . . 11
- 3.6 Intrusion Prevention . . . 12
- 3.7 Sonstiges . . . 12

**4 Implementierung** **13**
- 4.1 Auf dem Raspberry Pi . . . 13
- 4.2 Auf dem Server . . . 13

**5 Fazit** **14**
- 5.1 Evaluation der Arbeit . . . 14
- 5.2 Blick in die Zukunft . . . 14

**6 Anlagen** **15**

# 1 Einleitung

## 1.1 Motivation

Unser Ziel war es, ein Gerät zu entwerfen, welches einen oder mehrere Server auf Angriffe untersucht und diese eventuell sogar schon vorbeugt. Das Gerät sollte dabei möglichst einfach einzubauen sein, ohne große Änderungen am bestehenden System vornehmen zu müssen.

## 1.2 Code Repository

Unseren Code und Dokumentation haben wir auf github verwaltet. Hier der Link zu dem öffentlichen repository.

# 2 Anforderungen

Im folgenden sind die Anforderungen beschrieben, die wir uns für unser Projekt gesetzt haben.

## 2.1 Unabdingliche Anforderungen

### 2.1.1 File Based Intrusion Detection

Ein erfolgreicher Angriff hinterlässt spuren im System. Diese können sehr gut versteckt werden, etwa im Code von bestehenden Anwendungen.

Das Tool "AIDE" kann solche Angriffe jedoch aufdecken. Da dies einen sehr großen Teil von ausgeführten Angriffen erkennen kann, war dies uns sehr wichtig.

### 2.1.2 Network Intrusion Detection

Bevor ein Angriff lokale Dateien verändern kann, muss der Angreifer (meistens) erst über das Netzwerk gehen. Wird der Angriff schon dort erkannt, kann dieser schon schneller verhindert werden, und so etwa der Raub von Daten oder anderem verhindert werden. Aus diesem Grund war es uns sehr wichtig, auch diesen Teil mit unserem Projekt abzudecken.

### 2.1.3 Benachrichtigungen

Wurde ein Angriff entdeckt, sollte gehandelt werden. Was genau zu tun ist, ist oft äußerst komplex und individuell, dazu braucht es einen erfahreneren Systemadministrator. Damit dieser aber von dem Problem erfährt, ist es wichtig, dass dieser über verdächtige Aktionen benachrichtigt wird.

## 2.2 Optionale Anforderungen

### 2.2.1 Oberflächen

**Evebox** Evebox ist ein Suricata Alert- und Eventmanagement Tool für die Suricata IDS/NSM-Engine. Wir haben das Tool erweitert, um auch die Warnungen von AIDE anzeigen zu können. [2]

Viele Webseiten, über die schädliche Handlungen geschehen sind bekannt. Mit Pi-Hole kann verhindert werden, dass der Server überhaupt auf diese zugreift. Das fällt unter Intrusion Prevention.

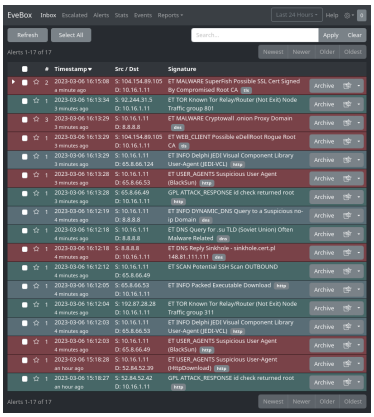


Abbildung 2: EveBox Oberfläche

**Pi-Hole** Viele Website, über die schädliche Handlungen geschehen sind bekannt. Mit Pi-Hole kann verhindert werden, dass der Server überhaupt auf diese zugreift. Sei es durch Manipulation oder sozial Engineering. [3]

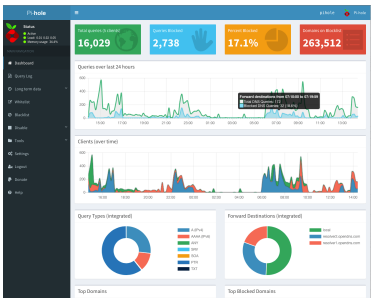


Abbildung 3: PiHole Oberfläche

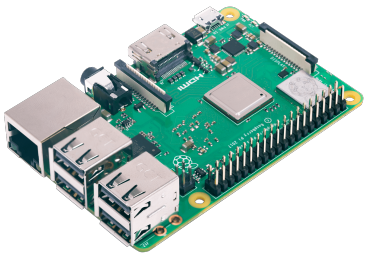

Abbildung 4: Darstellung eines Raspberry Pi

# 3 Verwendete Technologien

## 3.1 SSH

SSH bietet eine einfache und sichere Verbindung. Deswegen nutzen wir diese, um Befehle auf dem Server von unserem Gerät auszuführen. Zudem wird SSH zum Übertragen von Dateien eingesetzt.

## 3.2 Raspberry Pi

Ein Raspberry Pi ist ein ARM-basierter Einplatinencomputer mit einem Ein-Chip-System von Broadcom. Er ist klein und weit verbreitet, deswegen bietet er sich sehr gut an, um die Basis für unser Projekt zu sein. [4] Auch Technologien wie Pi-Hole waren darauf einfach umzusetzen.

## 3.3 File Based Intrusion Detection: AIDE

AIDE kann Veränderungen in Dateien und Ordnern entdecken, die sonst untergehen könnten. Außerdem erkennt es neue oder gelöschte Dateien und Ordner. Dies geschieht, indem Hashwerte für ausgewählte Ordner (und ihre Inhalte) erstellt werden und in einer Datenbank abgespeichert werden. Zu einem späteren Zeitpunkt kann dann mit dieser verglichen werden und Veränderungen können entdeckt werden. Wie nutzen wir dies?

Vorab der kompakte Ablauf, später mehr Details und Herausforderungen und ihre Lösungen.

AIDE läuft bei uns als Cronjob jeden Morgen um 2:00. Bei der ersten Ausführung von Aide auf einem Host wird eine Datenbank erstellt. Diese enthält Haswerte für Dateien und Ordner. Bei jeder weiteren Ausführung wird auch eine Datenbank für dem momentanen Stand erstellt und mit der vor 24 Stunden verglichen. Die erkannten Veränderungen werden dann geloggt und per Mail versand, beziehungsweise im JSON-Format in eine Logdatei geschrieben, welche Evebox darstellt. [5]

**Ausführung von AIDE** Die Berechnung der Hashwerte ist aufwendig. Deswegen sollte dies zu einem Zeitpunkt gemacht werden, in dem der Host Res-

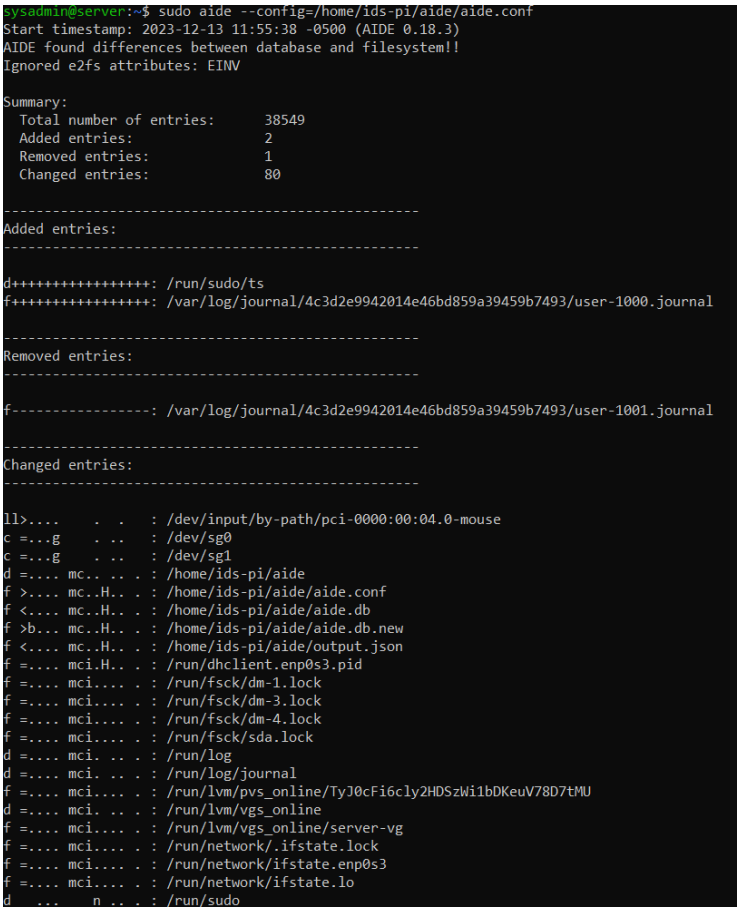


Abbildung 5: AIDE Konsolenausgabe von uns

sourcen zur Verfügung hat. Der Raspberry startet deswegen jeden Tag ab 2:00 die Erstellung der Datenbanken auf den Hosts.

**Persistenz der Datenbanken und Konfigurationsdateien** Die Datenbanken werden auf dem Host erstellt, von diesem gehen wir jedoch als nicht sicher aus. Deswegen sollten hier Daten nicht gespeichert werden. Denn hätte ein Angreifer sich Zugriff auf den Host verschafft, könnte er dies ja wieder verschleiern, was wir zu verhindern versuchen. Deswegen werden alle Datenbanken und die Konfigurationsdateien auf dem Raspberry Pi gespeichert. Die letzte Datenbank und Konfigurationsdatei wird bei jedem Ausführen von AIDE wieder auf den Host hochgeladen um damit zu arbeiten. Um zu verhindern, dass der Angreifer die packages von AIDE verändert, verifizieren wir diese vor jeder Ausführung auf den jeweiligen Hosts und geben eine entsprechende Fehlermeldung zurück.

**Erstellen der Konfigurationsdateien** Mit Aide kann und sollte individuell festgelegt werden, welche Ordner betrachtet werden. Werden dies irgendwann viele Ordner, kann dieser Prozess sehr aufwendig und Zeitintensiv werden. Genau deswegen sollte hier eine gute Auswahl getroffen werden, besonders weil AIDE keine Priorisierung der Änderungen durchführt. Der Nutzer sieht also immer alle Änderungen. Werden also etwa log-files betrachtet, die sich oft Ändern und auch viele neue hinzukommen, flutet dies die Ausgabe und wichtige Änderungen könnten übersehen werden. Sehr gut eignen sich deswegen ausführbare Dateien. Diese Ändern sich selten, etwa nur bei Updates. Außerdem sind diese ein sehr interessantes Ziel für Angreifer.

**Versand per Mail** Um auch über die Änderungen informiert zu werden, lesen wir den Output von AIDE mit einem Python-Skript aus und passen diese an die Ausgaben von Suricata an, um hier gute Übersicht für den Nutzer zu schaffen. Die Events der letzten 24h werden dann per Mail an die hinterlegte Adresse versendet.

## 3.4 Network Intrusion Detection: Suricata

Um Angriffe frühzeitig zu erkennen, insbesondere bevor sie lokale Dateien verändern können, setzen wir auf Network Intrusion Detection (NID) mit Suricata. Suricata ist eine leistungsfähige Open-Source-Software, die Netzwerkverkehr analysiert und nach potenziell schädlichem Verhalten sucht. [6]

**Funktionsweise von Suricata** Suricata arbeitet auf der Ebene des Netzwerkverkehrs und überwacht den Datenfluss in Echtzeit. Dabei nutzt es verschiedene Methoden, um Anomalien oder verdächtige Aktivitäten zu identifizieren:

1. **Signature-based Detection:** Suricata verwendet vordefinierte Signaturen, um bekannte Angriffsmuster zu erkennen. Diese Signaturen werden


Abbildung 6: Logo Suricata

regelmäßig aktualisiert, um gegen die neuesten Bedrohungen gewappnet zu sein.

2. **Anomaly-based Detection:** Durch die Analyse von Netzwerkverhalten erkennt Suricata auch ungewöhnliche Muster, die auf potenzielle Angriffe hindeuten können. Dies ermöglicht die Entdeckung neuer oder sich entwickelnder Bedrohungen.

3. **Protocol Detection:** Suricata erkennt und überwacht verschiedene Netzwerkprotokolle, um Abweichungen von den erwarteten Standards zu identifizieren.

**Integration in die bestehende Infrastruktur** Bei der Integration in die bestehende Infrastruktur eines Netzwerks gab es zahlreiche Herausforderungen. Suricata sollte den gesamten Datenverkehr des Servers analysieren können, jedoch sollte der Server nicht von dem Raspberry als Knotenpunkt abhängig sein. Hierzu hat unsere Gruppe drei verschiedene Herangehensweisen erarbeitet:

1. **Direkte Verbindung zum Server aus dem Netzwerk** [7]
   - Pro: Es ist nicht notwendig auszuwählen, welchen Server wir verwenden, da dies durch die IP-Adresse festgelegt ist.
   - Kontra: Das IDS wird keinen Verkehr sehen und würde nur den ausgehenden Verkehr scannen =>nicht wirklich hilfreich.
   - Idee: Verwendung von IPTables, um den gesamten Ein- und Ausgangsverkehr zunächst über den Pi zu routen?
2. **Verwendung des Pi als "Proxy":** Den Raspberry Pi als Gateway nach Außen und von außen sein IP-Adresse freigeben. [8]
   - Pro: Ich weiß, wie es geht, und es könnte wirklich einfach sein.
   - Kontra: Wie kann man feststellen, welche Verbindungen für den Pi und welche für den Server sind (zum Beispiel: SSH-Verbindung zum Pi oder zum Server)?
   - Kontra: Wie kann man feststellen, welchen Pakete an welchen Server weitergeleitet werden sollen, wenn wir mehrere verbundene Server haben?
   - Idee: Pi als Reverse Proxy verwenden?
3. **Erstellen eines eigenen Subnetzes, hinzufügen einer Route zu diesem Netzwerk auf dem Router, um über den Pi darauf zuzugreifen.** [9]
   - Pro: Könnte sehr einfach sein.
   - Pro: Leicht festzustellen, auf welchen Server zugegriffen werden soll.
   - Kontra: Wir haben nur eine LAN-Schnittstelle (normalerweise, wenn wir keinen zusätzlichen Adapter verwenden möchten).
   - Idee: Einem Adapter mehrere Adressen zuweisen und leiten den gesamten Verkehr zu diesem Subnetz weiterleiten.

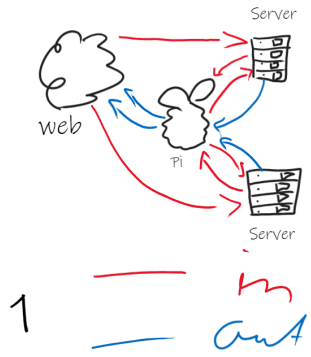


Abbildung 7: Idee 1: Direkt

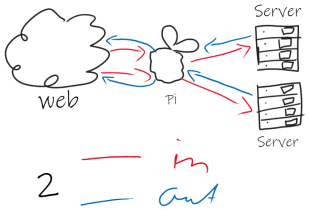


Abbildung 8: Idee 2: Proxy

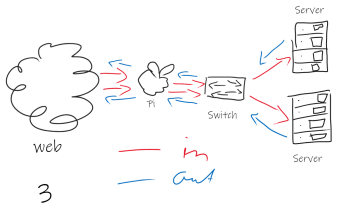


Abbildung 9: Idee 3: Eigenes Subnetz

Unsere gruppe entschied sich für die erste Variante und nutzte die IPTables Erweiterung ‘tee‘ um eine Kopie aller eingehender Pakete an den PI zu senden. Selbst wenn dieser ausfällt, arbeitet der Server unbeeinflusst weiter.

**Versand per Mail** Um über sogenannte Alerts informiert zu werden, lesen wir, ähnlich wie bei AIDE auch, die Lg-Dateien von Suricata mittels eines Python-Skriptes aus und generieren eine E-Mail. Die Events der letzten 24h werden dann per Mail an die hinterlegte Adresse versendet.

## 3.5 E-Mail

Damit ein Systemadministrator Standortunabhängig benachrichtigt werden kann, sind Mails ein sehr guter Weg. Hier gibt es auch schon sehr viele gegebene Möglichkeiten, diese über das Terminal zu versenden. Hier nutzen wir SSMTP. Um Mails zu versenden muss eine Textdatei erstellt werden, die eine bestehende Mail-Adresse und ein zugehöriges Secret enthält. Dann können Mails sehr einfach versendet werden. [10]


Abbildung 10: Screenshot E-Mail von IDS-Pi

## 3.6 Intrusion Prevention

Besser als einen Angriff zu entdecken, ist ihn zu verhindern. Die allermeisten Angriffe finden über das Internet statt und oft genug sind dabei bereits bekannte Webseiten im Spiel. Das Blocken von solch bekannten Webseiten kann Angriffe verhindern, bevor sie überhaupt stattgefunden haben. Dazu wird der Server so konfiguriert, dass er seine DNS-Auflösung über den Raspberry Pi abhandelt. Dieser kann mit der Technologie “Pi-Hole” [3] Websites blockieren. Für dieses gibt es umfangreiche Listen mit bekannten, potenziell schädlichen Websites und RegEx-Filter, welche beispielsweise URLS, welche auf ‘.exe’ enden oder nicht Lateinische Buchstaben (‘o’ U+006F‘ aus den Lateinischen Buchstaben und ‘o’ U+03BF‘ aus den griechischen Buchstaben) enthalten blockieren.

## 3.7 Sonstiges

- **IPTables:** Zusätzlich wurde zum Umgang mit dem Netfilter-Modul das Userspace-Programm IPTables verwendet. Dieses leitet beispielsweise alle eingehenden Pakete auf dem PI durch Suricata und sendet eine Kopie aller Pakete auf dem Server an den Pi.

- **Cron:** Der Cron-Deamon dient der zeitbasierten Ausführung von Prozessen. Dieser führt täglich die Überprüfung des Servers durch AIDE durch.

- **Samba:** Dient der zur Verfügung Stellung von Netzwerk-Ressourcen. Beim Teilen von Daten hierüber musste beachtet werden, dass die Samba-Ordner beschrieben werden können und potenziell auch überschrieben werden können. Die Daten, die der Raspberry davon braucht werden also in einen anderen Ordner verschoben.
- **systemd-resolved:** Dient dem manuellen Anpassen der '/etc/resolve.conf', welche den PI als DNS-Server deklariert.
- **Python:** Ist eine sehr mächtige und einfach zu erlernende Scriptsprache. In diesem Projekt, wird sie zum Verarbeiten von Daten verwendet, beispielsweise beim Vorbereiten der Mails und verarbeiten der AIDE-Log-Dateien.

# 4 Implementierung

## 4.1 Auf dem Raspberry Pi

Die Installation eines IDS-PI ist komplexer und nicht so automatisiert wie für einen server, muss aber auch deutlich seltener durchgeführt werden. Wichtig ist, dass der Zugang zu diesem geschützt ist, da secrets wie private ssh-keys für hochwertige Zugänge auf die server oder Tokens für Mail-Addressen auf diesem gepsichert werden.

Auf dem PI müssen bash-scripts und Konfigurationsdateien abgelegt werden, die wir erstellt haben. Aßerdem müssen cronjobs aufgesetzt werden, die diese ausführen. Für etwa die Installation von Suricata werden weitreichende Berechtigungen benötigt.

Den fertig eingerichteten IDS-Pi in das vorhandene Netz zu integrieren ist sehr einfach. Wenn er mittels LAN verbunden ist, versucht er sich automatisch eine IP mittels DHCP zu holen. Es wäre sinnvoll, ihm beim Router eine feste IP zuzuweisen, da, wenn sich die IP ändert, die hinzugefügten Server mittels AIDE und Pi-Hole nicht mehr überwacht werden. Er fängt automatisch an, alle ihm zugesendeten Pakete mittels Suricata zu analysieren und das Pi-Hole überwacht automatisch alle DNS-Anfragen, welche es bekommt. Zudem überprüft er alle 24 Stunden, ob neue Server für AIDE dazugekommen sind und initialisiert diese automatisch. Im Anschluss werden sie alle 24 Stunden mit AIDE überprüft.

## 4.2 Auf dem Server

Die Installation auf dem Server wurde mittels eines Installationsskriptes automatisiert. Hier ist es wichtig, dass der Server während der Installation dauerhaft mit dem Internet verbunden bleibt.

# 5 Fazit

## 5.1 Evaluation der Arbeit

Es war uns möglich sowohl alle Unabdinglichen, als auch alle Optionalen Anforderungen auf die Ein- oder andere Weise zu erfülle. Beispielsweise hatten wir anfangs vor, NIDS über das Pi-Hole laufen zu lassen, oder zu evaluieren, von welchen Dateien wir schon eine Hashsumme haben via Trusted Timestamping. Jedoch haben sich beiden Technologien im Laufe des Projekts für diesen Anwendungsfall als nicht geeignet herausgestellt.

Zudem haben uns vorgenommen, den IDS-Pi auf dem Server einfach und ohne große Änderungen auf dem Server als auch in der vorhandenen Infrastruktur vorzunehmen, zu integrieren. Leider ist uns das nur teilweise gelungen. Durch die Verwendung eines automatisierten Skriptes ist es sehr einfach den Server mit dem IDS-Pi zu verbinden. Auch die Installation eines (fertig eingerichteten) IDS-Pi in einem vorhandenen System ist sehr einfach (Plug and Play) möglich, jedoch verwenden wir sehr viel zusätzliche Software auf dem Server, welche teilweise sehr tief in das System eingreift (Beispiel: IPTables). Auch das Skript ist zwar voll funktionsfähig, jedoch toleriert es keine Fehler (Beispielsweise wenn die Internetverbindung während der Installation abbricht) und nach fehlerhafter Ausführung ist es sehr mühsam, das Skript fortzusetzen oder die Änderungen rückgängig zu machen.

## 5.2 Blick in die Zukunft

Wir möchten die Einrichtung des IDS-Pi mittels eines Debian-Packages vereinfachen. Dieses könnte man mittels des APT Paketmanagers installieren und hätte einen voll funktionsfähigen IDS-Pi.

Auch in den Bug bei den Netfiltern würden wir uns gerne tiefer einarbeiten und eventuell eine praktikable Lösung finden.

Des weitern wäre hier auch sehr nützlich, wenn anhand der erkannten Daten auch automatisch gehandelt werden kann. Etwa bestimmte IP-Addressen zu blockieren, oder den Server herunterzufahren.

Zuletzt haben wir uns angeschaut, wie man den Raspberry mittels eines Read-Only-Dateisystems und IPTables regeln robuster machen könnte.

# Abbildungsverzeichnis


1 Titelbild von Bing . . . . . 1
2 EveBox von Evebox . . . . . 4
3 PiHole von Wikimedia . . . . . 4
4 Raspberry Pi Darstellung von Reichelt . . . . . 5
5 AIDE Konsolenausgabe . . . . . 6
6 Logo Suricata von Suricata . . . . . 8
7 Idee 1: Direkt . . . . . 10
8 Idee 2: Proxy . . . . . 10
9 Idee 3: Eigenes Subnetz . . . . . 11
10 Screenshot E-Mail von IDS-Pi . . . . . 12


# 6 Anlagen

Eigenständigkeitserklärung

Ich erkläre mit meiner Unterschrift, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen dieser Arbeit, die dem Wortlaut, dem Sinn oder der Argumentation nach anderen Werken entnommen sind (einschließlich des *World Wide Web* und anderer elektronischer Text- und Datensammlungen), habe ich unter Angabe der Quellen vollständig kenntlich gemacht.

Ort, Datum Unterschrift

Eigenständigkeitserklärung

Ich erkläre mit meiner Unterschrift, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen dieser Arbeit, die dem Wortlaut, dem Sinn oder der Argumentation nach anderen Werken entnommen sind (einschließlich des *World Wide Web* und anderer elektronischer Text- und Datensammlungen), habe ich unter Angabe der Quellen vollständig kenntlich gemacht.

Ort, Datum Unterschrift