Uptimes

Mitgliederzeitschrift der
German Unix User Group

ISBN 978-3-86541-489-2

Proceedings

Frühjahrsfachgespräch 2013

in Frankfurt

guug

2013-1

lehmanns media

Donnerstag, 28. Februar – Vortragsprogramm

Anmeldung	09:00	**Anmeldung**
Keynote:	09:15	**Keynote:**
	von Kurt Garloff	
Kaffeepause	10:15	**Kaffeepause**
MySQL HA: Galera in the house (Erkan Yanar)	10:45	Ich habe eine WAF – Hilfe, sie loggt! (Christian Bockermann)
Oracle Migration – Hürden und Best Practices in einer hochverfügbaren Umgebung (Andrea Held)	11:30	Admins schlagen zurück: SSH-Angreifern mit Honeypots über die Schulter schauen (Andreas Bunten et al.)
Mittagspause	12:15	**Mittagspause**
Puppet, klar. Und dann? (Thomas Gelf)	13:45	DNS Rate Limiting (Matthijs Mekking)
Ansible – Verteilen, konfigurieren, machen (Philipp Grau)	14:30	Home Network Horror Stories (Michael Messner)
Kaffeepause	15:15	**Kaffeepause**
Aktuelle Entwicklungen bei Icinga und ein Ausblick auf Icinga2 (Bernd Erk)	15:45	Best of Audit 2012 oder „IT-Sicherheit? Evaluieren wir gerade.“ (Alexander Kodermann)
Netzmanagement mit HIRN (Robin Schröder)	16:30	IPv6 und Datenschutz – eine technische und juristische Bewertung (C. Wegener, J. Heidrich)
System- und Netzwerküberwachung mit Zabbix (Stefan Gazdag)	17:15	Die größten IPv6-Marketinglügen hinterfragt (Marc Haber)
Geselliger Abend	19:00	**Geselliger Abend**

09:15	**Keynote:** von Kurt Garloff
10:15	
10:45	PDF Debugging (vorwiegend) auf der Kommandozeile (Kurt Pfeifle)
11:30	strace – für BASH-Versteher (Harald König)
12:15	
13:45	TacNET – Grafische Verwaltung komplexer virtueller KVM Netze (Ralf Spenneberg)
14:30	Alternativen zur klassischen Virtualisierung (Oliver Rath)
15:15	
15:45	Sicheres und unabhängiges Datensharing mit ownCloud (Christian Schneemann)
16:30	Die eigene Cloud mit OpenStack Folsom (Martin Gerhard Loschwitz)
17:15	Aufbau einer IaaS-Umgebung mit OpenStack (Christian Berendt)
19:00	**Geselliger Abend**

Freitag, 1. März – Vortragsprogramm

Anmeldung	09:00	**Anmeldung**
Samba 4.0: Active Directory – und viel mehr (Michael Adam)	09:15	Best Practice für stressfreie Mailserver (Peer Heinlein)
ctdb performance bottlenecks and how to solve them (Volker Lendecke)	10:00	Strukturiertes Logging mit rsyslog (Rainer Gerhards)
Kaffeepause	10:45	**Kaffeepause**
Lustre/ZFS – verteiltes Dateisystem auf neuen Sohlen (Daniel Kobras)	11:15	Reboot reloaded – Linux-Kernel patchen ohne Reboot! (Udo Seidel)
Nahtlos skalierbares Storage mit Ceph (Martin Gerhard Loschwitz)	12:00	Kernelhänger mit System (oder „Systemüberwachung testen durch Fehlerinjektion“) (Stefan Seyfried et al.)
Mittagspause	12:45	**Mittagspause**
Btrfs und ZFS: eine Gegenüberstellung von Dateisystemen der neuen Generation (L. Grimmer, U. Gräf)	14:00	Linux ARM-Server im Unternehmenseinsatz (Daniel Gollub)
Illumos, SmartOS, OpenIndiana: Morgenröte für OpenSolaris oder Sonnenfinsternis? (Volker A. Brandt)	14:45	BOF „Was wollen wir mit der GUUG erreichen?“
Kaffeepause	15:30	**Kaffeepause**
Web-Performance-Optimierung mit varnish (Stefan Neufeind)	16:00	Interrupt-Routinen: Mit Störungen umgehen (Karsten Schulz)
Managing Loadbalancers on an Enterprise Level (Jan Walzer)	16:45	Erfolgreich selbständig sein in IT-Projekten oder: Wie frei ist ein Freelancer (Martina Diel)

Inhaltsverzeichnis

Frühjahrsfachgespräch 2013

Programmkomitee

Ich habe eine WAF - Hilfe, sie loggt!

Christian Bockermann
Lehrstuhl 8, Informatik, TU-Dortmund
Joseph-von-Fraunhofer Straße 23
44227, Dortmund
Deutschland
<christian.bockermann@udo.edu>

Zusammenfassung

Der Einsatz von Web Application Firewalls (WAF), zur Erkennung und Filtern von Angriffen auf der Anwendungsprotokollschicht ist eine der Maßnahmen, um Angriffe auf Web-Anwendungen zu unterbinden. Die dabei anfallenden Log- und Audit-Daten sind deutlich komplexer als herkömmliche, zeilenbasierte Log-Daten.

Dieser Artikel stellt mit der AuditConsole eine freie Server Software zur Zentralisierung und dem Management von Audit-Daten vor, die von Open-Source WAF Systemen wie ModSecurity oder IronBee erzeugt werden.

1 Motivation

Web-Anwendungen und Web-Services haben sich in den letzten Jahren zur zentralen Schnittstelle zwischen Nutzern im Internet und den Geschäftsprozessen der Anbieter entwickelt. Dabei sind diese Anwendungen zu einem lohnenswerten Ziel von Hackerangriffen geworden – neben der permanenten Erreichbarkeit von Web-Diensten stehen Web-Umgebungen häufig ungeschützt in vorderster Front der IT-Infrastruktur. Angriffe auf Web-Schnittstellen lassen sich zudem oft sehr leicht automatisiert durchführen - etwa durch SQL Injection Scanner, die Web-Anwendungen systematisch nach Verwundbarkeiten durch die Injektion von SQL-Befehlen durchsuchen. Web-Angriffe finden auf Ebene der Anwendungsschicht statt und sind daher von netzwerkbasierten Filtern wie IP-Firewalls nicht erkennbar.

Die Erweiterung von Firewalls um die Inspektion der Anwendungsschicht, also im Bereich von Web-Anwendungen, z.B. dem HTTP Protokoll, hat eine Reihe *Web Application Firewall* Systeme hervorgebracht. Diese Systeme fungieren oftmals als umgekehrter Proxy vor dem Web-Server und enthalten zumeist komplexe Regelsysteme um Angriffe auf Basis des HTTP-Protokolls zu erkennen. Eine der bekanntesten Open-Source Firewall Systeme stellt das *ModSecurity* Projekt [9, 10] bereit, das als Modul für den Apache Web-Server entwickelt wurde und inzwischen auch für andere Web-Server (z.B. NGinx, MicroSoft IIS) verfügbar ist.

Mit der komplexen Filterung der HTTP Verbindungen durch eine Web Application Firewall geht jedoch ein ebenso komplexes Aufkommen an Log- und Audit-Daten einher. Die meisten WAF-Systeme erlauben neben einer einfachen Alarm-Meldung je HTTP-Anfrage zusätzlich die Protokollierung der vollständigen HTTP Verbindungen, was für die genaue Anpassung der Regelsysteme und eine Analyse der erkannten Angriffe von immenser Bedeutung ist.

In diesem Artikel stellen wir mit der *AuditConsole* eine freie, Java-basierte Web-Anwendung vor, die Log- und Audit-Daten des *ModSecurity* WAF-Moduls verarbeitet, indiziert und archiviert. Neben der Speicherung von Audit-Daten integriert die AuditConsole ein benutzerorientiertes Rechtesystem um Daten mehrerer Web-Sites zu verarbeiten und Benutzern gleichzeitig nur den Zugriff auf die Daten der ihnen zugeordneten Web-Sites zu ermöglichen. Mit Hilfe einer einfachen Filter-Sprache lassen sich die Audit-Daten effizient durchsuchen und durch *Tags* markieren. Darüber hinaus stellt die AuditConsole eine eigene Reporting-Engine zur Verfügung, die die Erstellung von Berichten basierend auf den empfangenen Audit-Daten ermöglicht.

Der Artikel ist folgendermaßen gegliedert: In Abschnitt 2 geben wir am Beispiel des *ModSecurity* WAF-Moduls einen detailierteren Überblick über Web Application Firewalls und die anfallenden Audit-Daten. Zudem betrachten wir einige grundlegenden Aspekte des Log-Daten Managements, die aus unserer Sicht bei der Verarbeitung von Audit-Daten unterstützt werden müssen. Danach stellen wir in Abschnitt 4 die Architektur der *AuditConsole* vor und zeigen in Abschnitt 5, wie die in 2 definierten Aspekte mit Hilfe der AuditConsole unterstützt werden können.

2 Web Application Firewalls und Auditing

Wie in der Einführung bereits beschrieben, werden *Web Application Firewall* Systeme eingesetzt um Angriffe innerhalb der Anwendungsschicht zu erkennen und gefährliche Anfragen zu blockieren. Die

Systeme fungieren zumeist als *Reverse Proxy* Systeme, die stellvertretend für den Web-Server den Verbindungsendpunkt für Clients darstellen und Anfragen, nach deren Überprüfung durch ein Regelsystem, an den eigentlichen Web-Server weiterleiten. Das grundlegende Vorgehen von WAF-Systemen läßt sich im Wesentlichen durch die folgenden Punkte charakterisieren:

1. Annehmen der Client Verbindung
2. Parsen und Normalisieren der Anfragen
3. Überprüfen der Anfragen gegen ein Regelwerk, weiterleiten an Applikations-Server
4. Überprüfen der Antwort des Applikations-Servers und weiterleiten an den Client

Web Application Firewall Systeme zerlegen also Anfragen auf Basis des HTTP Protokolls und wenden in der Regel musterbasierte Regelwerke an, um Angriffe in diesen Anfragen zu erkennen. Dabei werden nicht nur die eingehenden Anfragen, sondern oftmals auch ausgehende Antworten des Applikations-Servers überprüft, z.B. um eventuelle Fehlermeldungen von Datenbanken zu unterbinden, die einem Angreifer zusätzliche Informationen über die Web-Umgebung liefern könnten.

Die Hauptanwendungsziele eines Web Application Firewall Systems liegen in erster Linie auf einer Überwachung des HTTP Datenverkehrs anhand bekannter Angriffsmuster. Darüber hinaus bieten diese Systeme die Möglichkeit des *virtual patching*, d.h. gefundene Schwachstellen in einer Anwendung können durch Regeln im WAF-System sehr schnell abgesichert werden. Dies ersetzt natürlich nicht die Behebung der Fehler in der Web-Anwendung selbst, ermöglicht jedoch eine schnelle Reaktionszeit auf Fehler, da Software-Änderungen häufig erst durch Qualitätssicherung und Tests abgenommen werden müssen. Dies gilt insbesondere bei Software von externen Anbietern, bei denen ggf. keine schnelle Fehlerbehebung möglich ist.

Im PCI Standard (Payment Card Industry) wird zudem für alle öffentliche zugänglichen Web-Anwendungen zwingend ein regelmäßiger Code-Review der Anwendungen oder alternativ der Einsatz einer Web Application Firewall vorgeschrieben [3].

2.1 ModSecurity - eine freie Web Application Firewall

ModSecurity ist eine Open-Source Implementierung einer Web Application Firewall als Modul für den weit verbreiteten Apache Web-Server [9]. Es erweitert den Apache Server um die Möglichkeit, Regeln zur Überprüfung von Anfragen in die Server-Konfiguration zu integrieren. In Kombination mit dem Proxy-Modul [6] läßt sich auf diese Weise eine Apache-basierte Web Application Firewall implementieren (vgl. Abbildung 1).

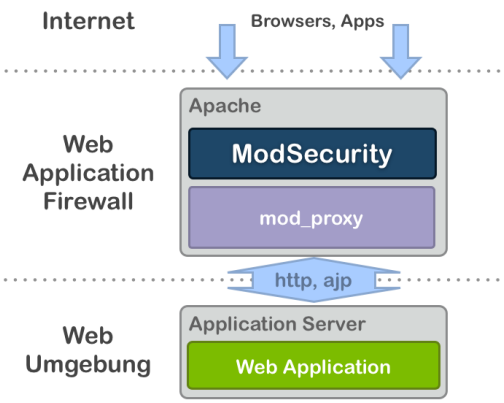


Abbildung 1: Ein Apache-Server mit *ModSecurity* als Reverse Proxy vor einem Applikations-Server.

Das *ModSecurity* Modul selbst implementiert jedoch lediglich die *Rule Engine*, die eine Reihe von Regeln auf die zu filternden Anfragen anwendet. Die Regeln werden als Bestandteil der Apache Konfiguration geschrieben und bestehen im Wesentlichen aus einem zu überprüfenden *Target*, einem *Operator*, einem *Wert* und einer Menge von Aktionen, die ausgeführt werden, wenn der Operator für die zu überprüfende Anfrage ein positives Ergebnis liefert. Als Beispiel sei die folgende Regel (Abbildung 2) gegeben, die als *Target* sämtliche Parameter (`ARGS`) einer Web-Anfrage überprüft. Der verwendete Operator `@rx` wendet dabei den gegebenen Wert als regulären Ausdruck an. Für den Fall, dass einer der Parameter die

```
SecRule ARGS "@rx alert" \
   log,msg:'Found "alert" in request param - possible XSS?',\
   deny,status:500,id:1001
```

Abbildung 2: Beispiel für eine *ModSecurity* Regel.

Zeichenkette `alert` enthält, wird die Anfrage blockiert und der Fehlerstatus 500 (*internal error*) and den Client gesendet. Die angegebene Meldung (`msg:`) wird dabei zusammen mit der Regel-ID 1001 im *error-log* des Web-Servers protokolliert.

Die Erstellung von Regeln für eine effiziente und korrekte Erkennung von Angriffen ist sehr komplex und benötigt ein hohes Maß an Verständnis für die Angriffe selbst, sowie Abläufe bei der Verarbeitung von HTTP Anfragen. Das OWASP *ModSecurity Core-Rules* Projekt pflegt ein umfassendes Regelwerk fertiger Regeln für die Erkennung einer Vielzahl von bekannten Web-Angriffen [5]. Diese müssen jedoch an die Begebenheiten der zu schützenden Web-Anwendung angepasst werden um die Fehlalarmrate zu minimieren.

Audit-Logging in ModSecurity

Zusätzlich zu Meldungen im *error log* unterstützt *ModSecurity* noch das wesentlich umfangreichere *audit log*, welches das Protokollieren der vollständigen Anfrage sowie – sofern bereits generiert – der Antwort ermöglicht. Über das Schlüsselwort `auditlog` wird durch eine Regel, deren Bedingung auf eine Anfrage passt, die vollständige Protokollierung der Anfrage aktiviert. Das dabei erzeugte *audit event* enthält (abhängig von der Konfiguration) mehrere Abschnitte, von denen jeder bestimmte Informationen zu der auslösenden Anfrage enthält. In der Regel enthält ein solches *audit event* die vollständige HTTP Anfrage und Antwort und kann somit relativ umfangreich sein. Tabelle 1 listet die wichtigsten Abschnitte eines *audit event* auf. Ein Beispiel für ein vollständiges *audit event* findet sich in Abbildung 11 im Anhang. *ModSecurity* schreibt diese ausführlichen Ereignisse je nach Konfiguration in eine große Datei (im Modus *Serial*) oder legt für jedes Ereignis eine eigene Datei an (im Modus *Concurrent*).

Abschnitt	Beschreibung
A	Verbindungsdaten (IP, Port)
B	Client-Anfrage Header
C	Client-Anfrage Body
F	Server-Antwort Header
G	Server-Antwort Body
H	Regel Nachrichten
K	Liste der gefeuerten Regeln

Tabelle 1: Die wichtigsten Teile eines Audit-Log Eintrags.

Um die Audit-Daten zu zentralisieren, enthält *ModSecurity* das Programm `mlogc` über das die Audit-Daten im *Concurrent*-Modus über HTTP PUT Requests auf einen Log-Server hochgeladen werden können. Dabei übernimmt `mlogc` auch das lokale Spooling von Ereignissen, für den Fall dass der Log-Server temporär nicht erreichbar ist. Der Upload der Ereignisse über `mlogc` ist der offizielle Weg, um *ModSecurity* Daten an einen Log-Server zu schicken.

3 Log Management von Audit Daten

Es ist offensichtlich, dass die Audit-Daten von WAF-Systemen wie *ModSecurity* weit über herkömmliche Log-Daten hinausgehen – wird jede Anfrage auf diese Weise protokolliert, so enthält die entstehende Log-Datei eine Kopie der vollständigen HTTP Kommunikation zwischen Client und Server. Ohne eine Auswertung dieser Audit-Daten ist der Betrieb eines WAF-Systems relativ wirkungslos. Nur durch eine genaue Analyse der Audit-Logs ist eine Anpassung der Regeln an die jeweilige Web-Anwendung effektiv möglich. Zugleich hilft ein zentrales Log-Management, die Effektivität der WAF-Konfiguration fortlaufend zu überwachen und zu dokumentieren. Die zentralen Fragestellungen, die sich beim Betrieb einer Web Application Firewall ergeben, sind z.B.

- Läuft die Web-Anwendung wie erwartet? Wie sind die Häufigkeiten der HTTP Antwort Stati?
- Weshalb wurde eine Anfrage genau blockiert? Welche Regeln waren dafür verantwortlich? Ab welchem Anomaly-Wert sollten Anfragen geblockt werden?
- Wie viele Anfragen wurden in der letzten Woche/im letzten Monat erkannt?
- Welche Arten von Angriffen gab es? Was waren die häufigsten Angriffe? Wurden durch virtuelle Patches Angriffe abgewendet?

Je nach zu überwachender Web-Anwendung und Rahmenbedingungen gibt es eine Vielzahl weiterer Fragestellungen, z.B. über Anfragen pro Sekunde je HTTP Sitzung. Auch durchschnittliche Antwortzeiten können von Bedeutung sein (z.B. für die Einhaltung von SLAs).

3.1 Analyse von ModSecurity Audit-Daten

Die aufgezeichneten Audit-Daten des *ModSecurity* Moduls sind zu komplex um mit einfachen Werkzeugen wie `less` und `grep` schnell an die gewünschten Informationen zu gelangen. Zudem enthalten die Audit-Daten einige Informationen nur implizit, wie beispielsweise die `anomaly-score`-Werte die von den CRS Regeln bereitgestellt werden. So enthält der Abschnitt K eines *audit event* alle für die Anfrage ausgeführten Regeln – aus diesen lassen sich diese Score-Werte errechnen, obwohl der tatsächliche Wert nicht explizit in den Daten enthalten ist. Für eine sinnvolle Analyse der Daten ist also eine weitere Aufbereitung erforderlich.

Im Folgenden stellen wir einige Anforderungen an die Verwaltung von Audit-Daten auf, die sich aus der Art der Daten und deren Erhebung ableiten lassen.

Zentrale Speicherung Im Falle von WAF-Modulen wie *ModSecurity* fallen die Audit-Daten in der Regel lokal auf den Systemen an. Da moderne Web-Umgebungen in den meisten Fällen aufgrund von Lastverteilung aus einer Vielzahl von WAF-Installationen bestehen, ist eine zentrale Verwaltung der Daten erforderlich um die gewünschten Informationen auch systemübergreifend erheben zu können.

Automatisierte Event-Verarbeitung Kritische Web-Anwendungen sind üblicherweise stark frequentiert, was häufig mit einem hohen Aufkommen an Audit-Daten einhergeht. Eine bloße Speicherung führt daher zu Datenmengen die manuell nicht zu bewältigen sind. Für eine effiziente Bewältigung der Analyse der Audit-Daten ist daher die Implementierung von automatisierten Prozessen erforderlich. Dafür

ist insbesondere die Integration mit bestehenden Systemen (IP Firewalls, RBL-Systeme) hilfreich um beispielsweise auf Grundlage der Audit-Daten Client-Systeme in einer IP-Firewall zu blockieren.

Reporting Zur Dokumentation der Auswirkungen von Sicherheitsmaßnahmen und der Einschätzung einer allgemeinen Sicherheitslage, ist die (periodische) Erzeugung statistischer Auswertungen, z.B. in Form von Reports erforderlich. Da die Anforderungen an ein derartiges Reporting sich im Detail häufig start unterscheiden oder sich mit der Weiterentwicklung von Web-Umgebungen verändern, muß die Reporting Funktionalität flexibel anpassbar sein.

Sichere Speicherung Die Audit-Daten anthalten zusätzlich zu den Verbindungsdaten eine Reihe sehr sensitiver Daten wie Cookies (Session ID), Nutzerdaten (Kennwörter) usw. Um eine Gefährdung der laufenden Web-Anwendungen zu minimieren, sollten sämtliche Daten möglichst in verschlüsselter Form übertragen und gespeichert werden. Zudem ist auch nur die Speicherung der Daten umzusetzen, die nach geltendem Recht gespeichert werden dürfen.

Moderne *Security Information and Event Management* Systeme (SIEM) unterstützen eine große Vielzahl dieser Anforderungen. Leider sind die meisten SIEM Lösungen nicht geeignet um die umfangreichen Audit-Daten von WAF-Systemen wie *ModSecurity* oder *IronBee* [7] direkt zu verarbeiten.

4 Die AuditConsole

Die AuditConsole entstand als Projekt aus einigen Programmen und Parser-Bibliotheken um ModSecurity Audit-Daten zu verwalten. Das Ziel der AuditConsole ist eine möglichst vollständige Implementierung der zuvor beschriebenen Anforderungen. Dabei soll die modulare Architektur sicherstellen, dass derzeit noch nicht unterstützte Funktionen auf einfache Weise hinzugefügt werden können. Durch möglichst einfache Schnittstellen soll zudem ermöglicht werden, eigene Erweiterungen in die Verarbeitung von Audit-Daten zu integrieren.

Im folgenden Abschnitt 4.1 geben wir einen Überblick über den Aufbau der AuditConsole und den grundlegenden Datenfluß der Audit-Daten. Auf dieser Grundlage erläutern wir in 4.2 wie Daten innerhalb der AuditConsole gespeichert und indiziert werden.

4.1 Architektur und Datenfluss

Die AuditConsole ist eine Java Web-Anwendung, die auf Basis bekannter Frameworks wie Struts2, Hibernate, Spring-Security u.a. implementiert wurde [4, 8, 2]. Zentrale Komponenten innerhalb der AuditConsole sind der *Data Store* und die *Event Verarbeitung*. Eingehende Audit-Daten werden in einer Pipeline verarbeitet und im *Data Store* gespeichert. Abbildung 3 zeigt die wesentlichen Komponenten der AuditConsole.

Die externen Schnittstellen der Anwendung sind das Web-Interface sowie eine REST API und ein Receiver-Servlet, das Audit-Daten von ModSecurity entgegennimmt. Darüber hinaus enthält die AuditConsole mehrere interne Dienste, z.B. für die Bereitstellung von *Realtime Block Lists*, dem *Task Service* zur Ausführung von Batch-Prozessen oder einem *Mail Service* für E-Mail Benachrichtigungen.

Die AuditConsole integriert die angesprochenen Komponenten innerhalb einer J2EE Web-Anwendung die in jedem konformen Servlet-Container (z.B. Apache Tomcat, Jetty) gestartet werden kann. Durch den Aufsatz auf das Hibernate Persistenz-Framework werden zudem eine Vielzahl von Datenbanken (MySQL, PostGres, Oracle) unterstützt.

Event Processor Pipeline

Die *Event Verarbeitung* ist durch eine Pipeline von *Event Prozessoren* realisiert. Jedes eingehende Event durchläuft diese Pipeline und wird schließlich im *Data Store* gespeichert. Die Prozessoren realisieren

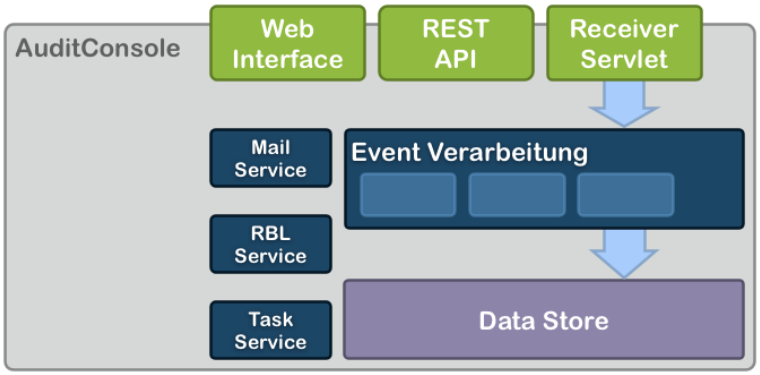


Abbildung 3: Grundlegende Architektur der AuditConsole.

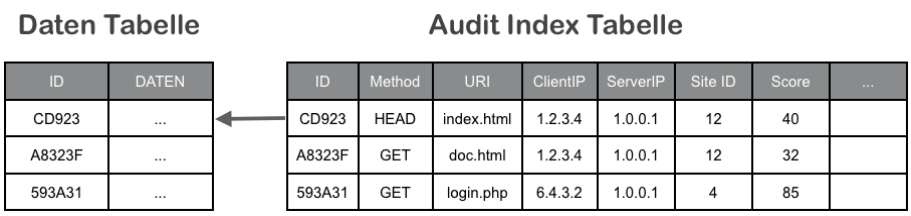
Funktionen wie die Auflösung von Load-Balancer Adressen, den Geo-IP Lookup für die Client IP oder die Berechnung des Anomaly-Scores. Abbildung 4 zeigt die wichtigsten Prozessoren in der Standard-Pipeline der AuditConsole.

Abbildung 4: Die Event Verarbeitung innerhalb der AuditConsole.

4.2 Indizieren und Speichern von Audit-Daten

Die empfangenen Audit-Daten werden von der AuditConsole im sogenannten *Data Store* abgelegt und indiziert. Der Data Store stellt einen abstrakten Speicher und Index für Audit-Daten dar. Die aktuelle Implementierung der AuditConsole verwendet dazu das Hibernate Persistenz Framework, was die Verwendung unterschiedlicher Datenbanken ermöglicht. Die Realisierung in Form eines abstrakten *Data Store*s ermöglicht zudem die Ersetzung des *Data Store* etwa durch eine verteilte Datenhaltung.

Der Datenbank-basierte Data Store speichert die Audit-Daten als Rohdaten in binärer Form in einer Datenbank-Tabelle. Zusätzlich wird jedes Ereignis anhand einer festen Liste von Variablen (z.B. `URL`, `REMOTE_ADDR`) charakterisiert und in einer Index-Tabelle abgelegt. Diese Index-Tabelle enthält einen Verweis auf das vollständige Datum in der Rohdaten-Tabelle.



ID	DATEN
CD923	...
A8323F	...
593A31	...

ID	Method	URI	ClientIP	ServerIP	Site ID	Score	...
CD923	HEAD	index.html	1.2.3.4	1.0.0.1	12	40	
A8323F	GET	doc.html	1.2.3.4	1.0.0.1	12	32	
593A31	GET	login.php	6.4.3.2	1.0.0.1	4	85	

Abbildung 5: Schematische Darstellung der Daten- und Index-Tabelle. Die genaue Realisierung enthält noch eine Reihe weiterer Tabellen zur effizienten Filterung der Daten.

Filter-Sprache für Audit-Daten

Die Index-Tabelle wird für die Filterung der Audit-Daten nach Benutzervorgaben benötigt. Dazu implementiert die AuditConsole eine einfache Filter-Sprache, die stark an die Syntax der *ModSecurity* Regeln angelehnt ist. Filter bestehen demnach aus einer *Variable*, einem *Operator* und einem *Wert*. Der folgende Filter selektiert beispielsweise die Ereignisse, für eine gegebene Client IP-Adresse:

```
REMOTE_ADDR @eq 192.168.10.4
```

Über die Syntax der *ModSecurity*-Regelsprache hinaus ermöglicht die Filter-Syntax der AuditConsole die Definition komplexerer Ausdrücke durch die logischen Operatoren `AND` und `OR`:

```
RULE_ID @eq 960018 AND AGE @lt 30min
```

Dieser leicht erweiterte Ausdruck definiert einen Filter für alle Audit-Daten, die von der Regel `960018` erzeugt wurden und vor *weniger als* (Operator `@lt`) 30 Minuten von der AuditConsole empfangen wurden. Eine umfassendere Dokumentation der Filtermöglichkeiten innerhalb der AuditConsole findet sich im *User Guide* [1].

Die Filter-Sprache wird in mehreren Teilen der AuditConsole verwendet um eine einheitliche Beschreibung der Daten zu ermöglichen. Für die Filterung mit Hilfe der Datenbank wurde ein Compiler geschrieben, der die Filter in die Hibernate Query Language (HQL) übersetzt und so über das Hibernate Framework auf Anfragen in der Datenbank abbildet.

4.3 Sites und User-Zugriff

Typischerweise werden WAF-Systeme als Reverse-Proxy für mehrere Web-Sites (etwa *virtuelle Hosts*) aufgesetzt. Dies führt dazu, dass eine Instanz der WAF häufig Audit-Daten für mehr als nur eine Web-Site protokolliert. Um die Daten der verschiedenen Web-Sites innerhalb der AuditConsole logisch voneinander zu trennen, können abstrakte *Sites* definiert werden. Eine *Site* fasst dabei die Ereignisse mehrerer Host- bzw. URL-Bereiche zusammen. So kann z.B. eine Site `jwall.org` für die Web-Adressen

`www.jwall.org`, `secure.jwall.org` und `download.jwall.org`

definiert werden. Die Zuordnung von Audit-Daten zu Sites wird über Regeln vom Administrator festgelegt und geschieht beim Empfang neuer Daten innerhalb des *Site-Mapping* Prozessors (vgl. Abbildung 4). So wird die oben genannte Zuordnung über die folgende Regel dargestellt:

```
REQUEST_HEADERS:Host @sx *jwall.org → jwall.org
```

Die Regeln entsprechen der gleichen Syntax wie der Filter-Sprache, die zur Abfrage der Datenbank verwendet wird (siehe 4.2). Hier ist `@sx` ein einfacher Wildcard-Operator, der den `Host`-Header der Anfrage auf den Ausdruck `*jwall.org` anwendet. Diese regelbasierte Zuordnung ermöglicht eine sehr flexible Gruppierung der Audit-Daten z.B. auch anhand der Server-Adresse, dem URL-Pfad, usw.

User Views

Das *Site*-Konzept bildet zugleich die Basis für die Zugriffsverwaltung von Benutzern auf Audit-Daten. Benutzer der AuditConsole müssen sich zunächst am Web-Interface authentifizieren. Die Authentifizierung geschieht über ein einfaches Benutzername/Kennwort-Login oder externe Mechanismen wie OpenID oder einen CAS-Server. Innerhalb der AuditConsole muß für jeden Benutzer zunächst ein *View* angelegt werden, der festlegt, auf welche *Sites* der Benutzer zugreifen darf. Ohne einen definierten *View* ist ein Benutzer nicht in der Lage auf irgendwelche Audit-Daten zuzugreifen.

Auf diese Weise realisiert die AuditConsole ein Mehrbenutzerkonzept bei dem Daten verschiedener Web-Sites nur von den jeweils berechtigten Benutzern eingesehen werden können. Gleichzeitig ermöglicht die Site-Zuordnung der Daten beim Empfang eine sehr effiziente Umsetzung des Mehrbenutzerkonzepts.

4.4 Tasks und die Reporting Engine

Die bisher vorgestellten Komponenten der AuditConsole ermöglichen eine interaktive Filterung von Audit-Daten und bieten über das Web-Interface eine übersichtliche Darstellung der Daten für den Benutzer. Ein wichtiger Punkt im Management von Log-Daten ist jedoch die automatisierte Weiterverarbeitung und Auswertung der Daten. Log-Daten stellen kontinuierliche Datenquellen dar, die bei einfacher Speicherung schnell auch die Kapazitäten des Log-Systems erreichen.

Für die automatisierte Verarbeitung der Daten integriert die AuditConsole einen *Task Service*, der es erlaubt periodisch Prozesse zu starten, die die anfallenden Daten aufräumen. So ist beispielsweise ein periodisches Löschen oder ein Export der Daten ein wichtiger Aspekt um die Datenbank nicht unendlich wachsen zu lassen. Zu diesem Zweck enthält die AuditConsole eine Reihe von definierten Tasks, wie z.B. der *Archivierung*. Dieser Task erlaubt es, alte Audit-Daten aus der Datenbank zu löschen und (falls erwünscht) zuvor in externe Dateien zu exportieren.

Erstellung von Berichten mit der Reporting Engine

Die Erstellung von Reports auf Grundlage von Audit-Daten ist wohl mit eine der wichtigsten Anforderungen beim Management von Log-Daten. Die AuditConsole enthält eine eigene, auf *DocBook* [11] basierende Reporting Engine. Dies erlaubt die Definition von Reports über XML Templates, entsprechend der Anforderungen an die Erstellung von Berichten. Ein Report-Template ist in dieser Form ein XML *DocBook* Dokument, das um zusätzliche Elemente erweitert wird. Der XML Auszug in Abbildung 6 zeigt ein einfaches Beispiel für ein *DocBook* Report-Template, das eine geographische Darstellung der Verteilung der Audit-Daten erzeugt und auf einer Weltkarte visualisiert.

```
<?xml version="1.0" encoding="UTF-8"?>
  <article version="5.0">
    <title>Beispiel-Report</title>
    <articleinfo>  <!-- Aus Platzgruenden weggelassen --> </articleinfo>
    <section>
       <title>Geographische Verteilung der Audit-Daten</title>
       <para>
          Die folgende Karte zeigt die Verteilung der Audit-Daten
          anhand der Geographischen Aufloesung von IP-Addressen.
       </para>
       <GeoMap address="REMOTE_ADDR" aspect="COUNT" />
    </section>
  </article>
```

Abbildung 6: Beispiel für ein Report-Template. Das `GeoMap` Element wird von der AuditConsole bereitgestellt und vom Report-Task berechnet. Die übrigen Elemente gehören zum *DocBook* Standard.

Zusätzlich zum Element `GeoMap` unterstützt die Reporting Engine die Erzeugung von Pie-Charts und Tabellen für nahezu beliebige Aspekte der Audit-Daten. Über die in Abschnitt 4.2 beschriebenen Filter lassen sich zudem sehr feingranular Teilmengen von Ereignissen in Unterabschnitten des Reports gesondert darstellen (z.B. nur für bestimmte URL-Präfixe).

Wurde ein Report-Template definiert kann die Erstellung des Reports über das Web-Interface angestoßen werden. Die Report-Erstellung läuft dann als Task innerhalb des *Task Service* der AuditConsole ab. Durch die Verwendung von *DocBook* als Grundlage für die Reporting Engine ist über die von *DocBook* bereitgestellten Stylesheets ein Export der Berichte als HTML, PDF und viele andere Formate möglich.

4.5 Automatische Event-Verarbeitung

Die in Abschnitt 4.1 beschriebene *event processor* Pipeline enthält unter anderem einen Prozessor der es erlaubt, Event-Regeln auf den empfangenen Audit-Daten auszuführen. Dazu hat jeder Benutzer der AuditConsole die Möglichkeit, Regeln zu definieren, die auf eingehenden Events Aktionen ausführen.

Als einfaches Beispiel ließe sich so etwa eine E-Mail Benachrichtigung für alle Ereignisse, deren Anomaly-Score (z.B. auf Basis des *Core Rules* Regelwerks) einen bestimmten Wert übersteigt, definieren. Auch diese Regeln basieren auf der einheitlich definierten Filter-Syntax. Als mögliche Aktionen, die aufgrund einer solchen Event-Regel ausgeführt werden können, unterstützt die AuditConsole derzeit die folgenden *rule actions*:

- *delete* – löscht das Ereignis (sofern der Benutzer die entsprechenden Rechte hat)
- *tag event* – Markiert das Ereignis mit einem benutzerdefinierten Tag
- *notify/send mail* – sendet eine E-Mail-Benachrichtigung mit Informationen über das Event
- *call URL* – ruft eine URL mit optionalen Parametern auf (z.B. einen REST-Service)
- *rbl block/unblock* – setzt die Client IP des Ereignisses für eine angegebene Zeit auf die RBL-Liste (oder entfernt sie).

Natürlich werden die Event-Regeln eines Benutzers nur für die Ereignisse ausgeführt, die auch im *View* des jeweiligen Benutzers enthalten sind.

5 Log-Management mit der AuditConsole

Die vorangegangenen Abschnitte haben einen Einblick in die Architektur und Bestandteile der AuditConsole gegeben. Mit der AuditConsole lassen sich die Audit-Daten für eine Vielzahl von *ModSecurity* Instanzen zentral speichern. Gleichzeitig stellt die AuditConsole die Möglichkeit für eine interaktive Exploration der Daten sowie deren automatisierter Verarbeitung (z.B. über Event Regeln) bereit.

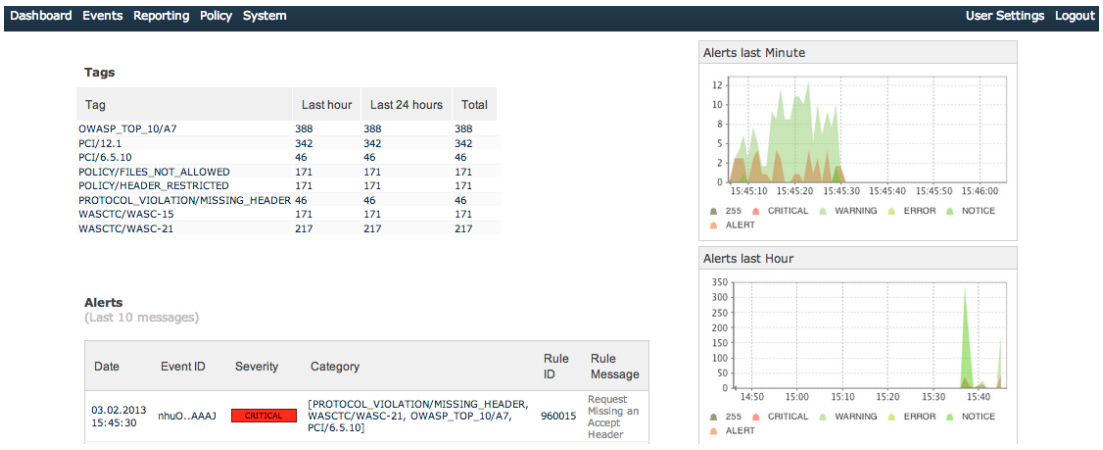


Abbildung 7: Das Dashboard der AuditConsole.

5.1 Web-Interface und Dashboard

Die Bedienung der AuditConsole erfolgt üblicherweise mit einem Browser über das Web-Interface. Diese Schnittstelle stellt neben einem Dashboard alle Funktionen zur Anzeige und Filterung von Audit-Daten

bereit. Über das Web-Interface kann zudem die Erstellung von Reports angestoßen werden und Regeln zur automatisierten Verarbeitung eingehender Audit-Daten definiert werden. Abbildung 7 zeigt das zentrale Dashboard der AuditConsole mit den Anzeigen der aktuell empfangenen Audit-Daten.

Neben dem Dashboard ist die Liste der Audit Ereignisse das wichtigste Element der AuditConsole. Über diese Liste lassen sich mit Hilfe der Filter-Syntax interaktiv gezielt die gewünschten Ereignisse anzeigen. Das Filtern nach Regel-IDs ermöglicht es beispielsweise, gezielt nach den Auswirkungen von Regeln in den Audit-Daten zu suchen. Abbildung 8 zeigt die Listen-Ansicht der Audit Ereignisse mit dem Filter `RESPONSE_STATUS @ge 400`.

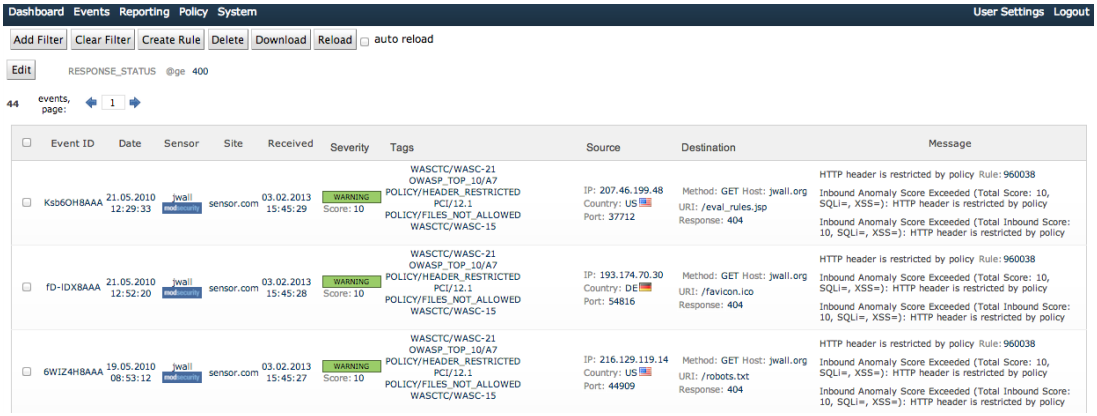


Abbildung 8: Die Event Listenansicht der AuditConsole.

Über die Liste lassen sich Ereignisse zudem mit beliebigen Schlüsselwörtern markieren. Dies ermöglicht es Benutzern z.B. Audit-Daten für Regressions-Tests zu markieren und diese für Änderungen von Regelwerken oder System-Updates aus der AuditConsole herunterzuladen.

5.2 Reports

Die AuditConsole stellt bereits einfache Report-Templates bereit. Mit Hilfe dieser Templates lassen sich direkt ein paar beispielhafte Berichte erzeugen. Abbildung 9 zeigt einen Ausschnitt aus einem Report der mit Hilfe des Beispiel XML-Templates aus Abschnitt 4.4 erzeugt wurde.

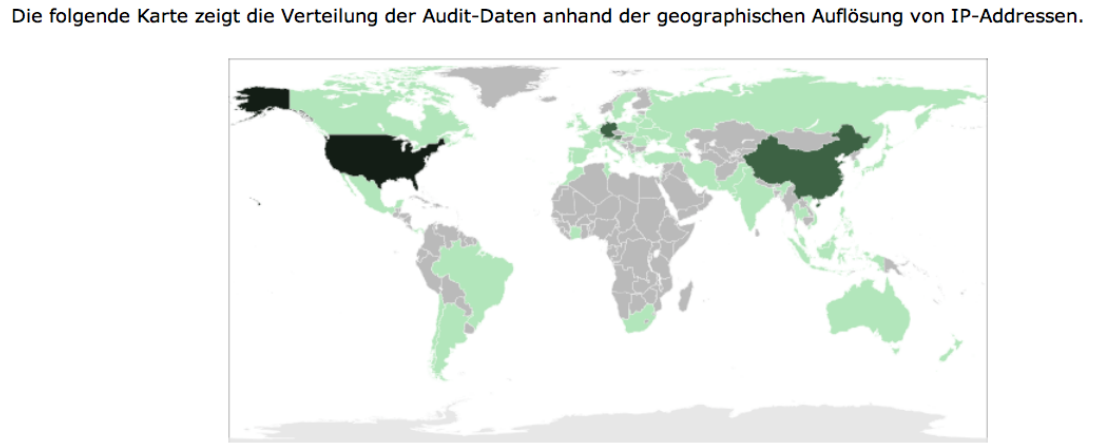
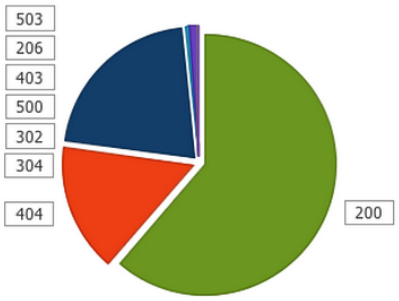


Abbildung 9: Ergebnis des Report-Templates aus Abschnitt 4.4.

Aufgrund der einfachen Anpassbarkeit der Templates ist es jedoch auch möglich gezielt Berichte für Fragestellungen zu Regeln oder der Häufigkeiten der HTTP Status Codes zu erstellen. Mit Hilfe des XML Elementes `Summary` läßt sich so auf einfache Weise der Report um eine Aufstellung der Status Codes erweitern. Das `Summary` Element ermöglicht die Aggregation von Ereignissen anhand unterschiedlicher Aspekte. Das nachfolgende XML-Beispiel zeigt das `Summary` Element für eine tabellarische Zusammenfassung von Ereignissen über deren Response-Status, gruppiert nach dem `Host`-Header der Anfragen. Auf diese Weise lassen sich die verschiedensten Aggregationen für unterschiedliche Aspekte in der

```
<section>
   <Summary groupBy="REQUEST_HEADERS:Host" aspect="RESPONSE_STATUS" />
</section>
```

Erstellung von Berichten verwenden. Abbildung 10 zeigt die Ausgabe der Status Codes im Report mit dem `Summary` Element.



REQUEST_HEADER:Host	Response Status Code							
	200	404	304	302	500	403	206	503
www.jwall.org	7564	1946	2624	51	124	1	19	2

Abbildung 10: Ergebnis des Report-Templates erweitert um das `Summary` Element.

6 Fazit und Ausblick

Für den Betrieb einer Web Application Firewall ist eine sorgfältige Betrachtung der entstehenden Audit-Daten unumgänglich. Mit der AuditConsole steht ein umfangreiches Werkzeug zur Zentralisierung und Exploration von Audit-Daten für die Open-Source Web Application Firewall *ModSecurity* zur Verfügung. Neben der Exploration von Ereignissen bietet die AuditConsole eine flexible Reporting Engine zur Erstellung von Berichten über die empfangenen Audit-Daten.

Die AuditConsole hat sich über die letzten Jahre zur Standard-Anwendung für die Verwaltung von *ModSecurity* Logs entwickelt und wird auch auf `www.modsecuriy.org` selbst, sowie in zahlreichen Projekten wie dem WASC Honeypot Projekt zum Log-Management eingesetzt.

Zudem wird die AuditConsole stetig weiterentwickelt. Neben der verschlüsselten Speicherung von Audit-Daten im *Data Store* ist auch eine verteilte Speicherung über mehrere *Data Stores* derzeit in der Entwicklung. Auf Basis der verteilten Speicherung läßt sich die AuditConsole so auch für die Verwaltung größer Web-Umgebungen einsetzen.

Literatur

[1] Christian Bockermann. AuditConsole - User Guide, 2012.

[2] Spring Source Community. Spring Framework.

[3] PCI Security Standards Council. Payment Card Industry (PCI) Data Security Standard, 2006.

[4] Craig R. McClanahan et. al. Struts - A Framework for Creating Java Web Applications, 2007.

[5] Ryan Barnet et. al. The ModSecurity Core Rule Set Project.

[6] Apache Foundation. Apache Module mod_proxy.

[7] Brian Rectanus, Ivan Ristic, Nick LeRoy, Nick Kew, Craig Forbes, Sam Baskinger, and Christopher Alfeld. IronBee Open Source Web Application Firewall, 2012.

[8] Hibernate Community Red Hat. Hibernate - Relational Persistence for Java and .NET, 2012.

[9] Ivan Ristic. ModSecurity - A Filter-Module for the Apache Webserver, 1998.

[10] Ivan Ristic. *ModSecurity Handbook*. Feisty Duck, 2010.

[11] Norman Walsh. Docbook 5: The Definitive Guide, 2010.

Anhang

```
--5f063027-A--
[20/Jun/2010:00:45:24 +0200] 08PHhH8AAAEAAEAgUu8AAAAL 220.181.7.78 12005 78.46.79.252 80
--5f063027-B--
GET / HTTP/1.1
Host: www.jwall.org
Accept-Language: zh-cn
Connection: close
User-Agent: Baiduspider+(+http://www.baidu.com/search/spider.htm)

--5f063027-F--
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Set-Cookie: JSESSIONID=5E0904023B343F302B02263CB9D7D4A5; Path=/
Connection: close
Transfer-Encoding: chunked

--5f063027-G--
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
....
</html>

--5f063027-H--
Message: Match of "rx ^OPTIONS$" against "REQUEST_METHOD" required. \
  [file "/opt/modsecurity/rules/crs-2.0.7/base_rules/modsecurity_crs_21_protocol_anomalies.conf"] \
  [line "46"] [id "960015"] [rev "2.0.7"] [msg "Request Missing an Accept Header"] [severity "CRITICAL"] \
  [tag "PROTOCOL_VIOLATION/MISSING_HEADER"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"]
Message: Operator GE matched 0 at TX:anomaly_score. \
  [file "/opt/modsecurity/rules/crs-2.0.7/base_rules/modsecurity_crs_49_inbound_blocking.conf"] \
  [line "18"] [msg "Inbound Anomaly Score Exceeded (Total Score: 5, SQLi=, XSS=): Request Missing an Accept Header"]
Message: Warning. Operator GE matched 0 at TX:inbound_anomaly_score. \
  [file "/opt/modsecurity/rules/crs-2.0.7/base_rules/modsecurity_crs_60_correlation.conf"] \
  [line "35"] [msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 5, SQLi=, XSS=): Request Missing an Accept Header"]
Apache-Handler: proxy-server
Stopwatch: 1276987524237188 45111 (418 1632 -)
Producer: ModSecurity for Apache/2.5.11 (http://www.modsecurity.org/); core ruleset/2.0.7.
Server: Apache/2.2.3 (CentOS)

--5f063027-K--
SecAction "phase:1,t:none,pass,nolog,initcol:global=global,initcol:ip=%{remote_addr}"
SecAction "phase:1,t:none,nolog,pass,setvar:tx.paranoid_mode=0"
SecAction "phase:1,t:none,nolog,pass,setvar:tx.inbound_anomaly_score_level=20"
SecAction "phase:1,t:none,nolog,pass,setvar:tx.outbound_anomaly_score_level=15"
SecAction "phase:1,t:none,nolog,pass,setvar:tx.critical_anomaly_score=20,setvar:tx.error_anomaly_score=15,\
    setvar:tx.warning_anomaly_score=10,setvar:tx.notice_anomaly_score=5"
SecAction "phase:1,t:none,nolog,pass,setvar:tx.max_num_args=255"
SecAction "phase:1,t:none,nolog,pass,setvar:'tx.allowed_methods=GET HEAD POST OPTIONS',\
   setvar:'tx.allowed_request_content_type=application/x-www-form-urlencoded ...
SecRule "REQUEST_METHOD" "@rx ^(?:GET|HEAD)$" \
   "phase:2,chain,rev:2.0.7,t:none,pass,nolog,auditlog,msg:'GET or HEAD requests with bodies',\
   severity:2,id:960011,tag:PROTOCOL_VIOLATION/EVASION,tag:WASCTC/WASC-21,tag:OWASP_TOP_10/A7,\
   tag:PCI/6.5.10,tag:http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.3"
SecRule "&REQUEST_HEADERS:Accept" "@eq 0" "phase:2,pass,chain,rev:2.0.7,t:none,nolog,auditlog,\
   msg:'Request Missing an Accept Header',severity:2,id:960015,tag:PROTOCOL_VIOLATION/MISSING_HEADER,\
   tag:WASCTC/WASC-21,tag:OWASP_TOP_10/A7,tag:PCI/6.5.10"
SecRule "REQUEST_METHOD" "!@rx ^OPTIONS$" "t:none,setvar:tx.msg=%{rule.msg},\
   setvar:tx.anomaly_score=+%{tx.notice_anomaly_score},setvar:tx.protocol_violation_score=+%{tx.notice_anomaly_score},\
   setvar:tx.%{rule.id}-PROTOCOL_VIOLATION/MISSING_HEADER-%{matched_var_name}=%{matched_var}"
...
--5f063027-Z--
```

Abbildung 11: Ein ModSecurity *audit event*, das eine vollständige Anfrage und Anwort enthält. Die Server Antwort (HTML Code) sowie einige Regeln wurden aus Platzgründen gekürzt.

Interrupt Routinen

Karsten Schulz
Lifehacker Methoden
Hauptfeld 18
44369 Dortmund
Deutschland
<k.schulz@lifehacker-methoden.de>

1 Unterbrechungen am Arbeitsplatz

Können Sie an Ihrem Arbeitsplatz jede Aufgabe von Anfang bis Ende wie geplant durchführen? Oder werden Sie dabei ab und zu unterbrochen? An den meisten Computerarbeitsplätzen sind Störungen und Unterbrechungen die Regel. Ob E-Mail, Instant Messenger, Kollege oder Chef. Unterbrechungen sind allgegenwärtig. Manchmal sind sie willkommen und manchmal stören sie. Unterbrechungen bei Arbeitsvorgängen sind ein wichtiges Thema in der Forschung. Untersuchungen haben zum Beispiel ergeben, dass viele Flugzeugunglücke deshalb passieren, weil der Pilot bei einer Routinetätigkeit unterbrochen wurde und er nach der Unterbrechung unbemerkt eine Teilaufgabe übersprungen hatte. Lesen Sie in diesem Artikel, wie Störungen die Arbeitsqualität des Computerarbeiters beeinflussen und wie die negativen Auswirkungen minimiert werden können. Am Ende des Artikels gibt es dazu nützliche Tipps für die Praxis.

2 Der Faktor Mensch

Der durchschnittliche Informationsarbeiter ist nach eigenem Bekunden ein Multitasker. Er liebt selbstbestimmtes Arbeiten und nutzt moderne Technologien für die Erledigung seiner Aufgaben. Dabei ist er jedoch biologischen Beschränkungen unterworfen. Sein Arbeitsgedächtnis und seine Aufmerksamkeit sind Ressourcen, die nicht unbegrenzt zur Verfügung stehen. Außerdem stellen sich regelmäßig Ermüdungs- und Erschöpfungszustände ein, die Einfluss auf die Qualität der Arbeitsergebnisse haben. Bei intensivem Multitasking gerät der Arbeitende schneller an diese Grenzen.

2.1 Multitasking

Forscher sind sich einig, dass es im Gehirn verschiedene Ressourcen gibt, die gleichzeitig arbeiten können. Es ist möglich, dass sensorische Prozesse gleichzeitig mit motorischen oder kognitiven Prozessen ablaufen. Wie gut das funktioniert ist individuell ausgeprägt und hängt von der jeweiligen Leistung des Arbeitsgedächtnisses, der Aufmerksamkeit und der fluiden Intelligenz ab.

Durch Automatisierung kann der Grad des Multitasking erhöht werden. Gewohnte und gut trainierte Arbeitsabläufe und Routinen benötigen zur Ausführung weniger mentale Ressourcen. Einem Experten fällt Multitasking leichter als einem Laien. Ein Experte zeichnet sich dadurch aus, dass er über ein assoziativ-operatives System verfügt, welches so differenziert und genau ist, dass er schnell und Ressourcen schonend handeln kann.

Die Bundesanstalt für Arbeit berichtet im Stressreport Deutschland 2012, dass Multitasking ein Stressauslöser sein kann. Die dauernden Unterbrechungen und Wechsel der Aufmerksamkeit führen zu schnellerer Ermüdung und Überlastung. Durch die resultierende Unkonzentriertheit steigt die Fehlerrate.

2.2 Aufmerksamkeit

Man geht davon aus, dass wir mit unserem Arbeitsgedächtnis dauernd bewusst an unsere Ziele denken müssen, damit wir sie nicht aus den Augen verlieren. Wenn ein Ziel nur zwei Sekunden nicht abgerufen wird, verliert es an Aktivierungsintensität. Wenn unsere Aufmerksamkeit abgelenkt wird, vergisst unser Gehirn buchstäblich die eigentliche Aufgabe mehr und mehr, bis das ablenkende Ereignis selbst eine so hohe Aktivierungsintensität hat, dass es unser Arbeitsgedächtnis vollends in Anspruch nimmt. In diesem Moment ist die ursprüngliche Aufgabe vergessen.

Zu dem Phänomen des Vergessens wird auch das in der Literatur als „prospective memory failure“ bezeichnete Vergessen der Erledigung zukünftiger Aufgaben gezählt. Wer hat es noch nicht erlebt: wir nehmen uns vor, etwas später zu erledigen, vergessen es dann aber doch.

3 Von Störungen und Unterbrechungen

Unterbrechungen werden durch Störungen ausgelöst. Eine Störung ist ein Ereignis, welches die Aufmerksamkeit von der aktuellen Tätigkeit abzieht.

3.1 Externe Störquellen

Störungen können durch externe oder interne Ereignisse verursacht werden. Externe Ereignisse sind zum Beispiel

- der Telefonanruf
- der Kollege, Chef oder Kunde
- akustische oder optische Signale (E-Mail, IM, . . .)

Wer beobachtet, dass seine Arbeit häufig durch externe Einflüsse gestört oder unterbrochen wird, der sollte untersuchen, welche der Störungen sich positiv und welche sich negativ auswirken. Positive Störungen sind meistens die, die als Anregung, Abwechslung oder Inspiration empfunden werden. Sie bereichern in der Regel das Arbeitsumfeld. Das Auftreten negativer Störungen sollte durch organisatorische und kommunikative Regelungen möglichst vermieden werden.

3.2 Interne Störquellen

Es ist sinnvoll sich auch seiner inneren Störquellen bewusst zu werden. Wenn Sie den aktuellen Arbeitsvorgang unterbrechen, um auf die Toilette zu gehen oder etwas zu trinken zu holen, dann ist das eine Störung, die nicht nötig gewesen wäre, wenn Sie sich um Ihre Bedürfnisse vor Aufnahme der Tätigkeit gekümmert hätten. Häufige Quellen für innere Störungen sind:

- Müdigkeit
- Unkonzentriertheit
- Hunger
- Durst
- Unlust

Seien Sie gut zu sich selbst und sorgen Sie dafür, dass diese inneren Faktoren möglichst keinen oder nur geringen Einfluss auf Sie haben, wenn Sie eine Ihrer Hauptaufgaben erledigen.

3.3 Unterbrechungen

Wenn die Störung so intensiv ist, dass die Hauptaufgabe nicht mehr fortgeführt wird, dann entsteht eine Unterbrechung. Bei einer Unterbrechung wird die Arbeit an der ursprünglichen Aufgabe eingestellt und eine andere Tätigkeit ausgeführt. Jetzt droht höchste Gefahr für Fehler. Die Wahrscheinlichkeit, dass Sie die Hauptaufgabe entweder überhaupt nicht mehr, falsch oder an einer falschen Stelle fortsetzen, ist sehr hoch.

Schon eine Unterbrechung von nur 3 Sekunden Dauer kann die Fehlerrate verdoppeln, wie Erik Altman von der Michigan State University feststellte.

4 Die Anatomie von Unterbrechungen

Betrachtet man den zeitlichen Ablauf einer Unterbrechung dann lässt sich folgende Struktur erkennen. Zunächst wird in der ursprünglichen Aufgabe eine Tätigkeit nach der anderen abgearbeitet. Im Schaubild sind die Tätigkeiten als „NA“, als „Nächste Aktion“ bezeichnet. Wenn eine Störung eintritt, beginnt die Phase der Unterbrechungsverzögerung. Mental sind wir noch bei der ursprünglichen Aufgabe, unsere Aufmerksamkeit wandert aber bereits zum störenden Ereignis.

Dann wird entschieden, dass die Unterbrechung bearbeitet wird. In dieser Phase verliert unser Arbeitsgedächtnis alle Informationen der ursprünglichen Aufgabe. Es bearbeitet die Unterbrechungsaufgabe.

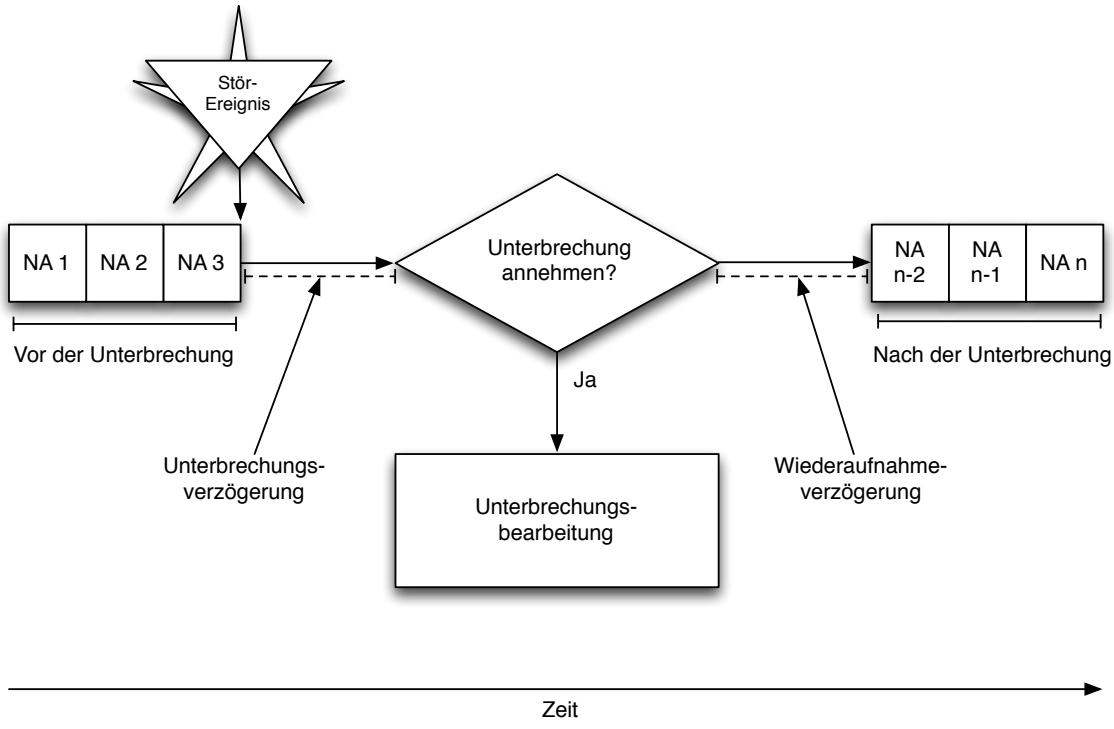


Figure 1: Ablauf einer Unterbrechung nach Brixey et al.

Irgendwann nach Erledigung der Unterbrechungsaufgabe erinnert sich unser Gehirn an die ursprüngliche Aufgabe. Bevor diese wieder aufgenommen werden kann, müssen wir mental umschalten und versuchen „den Faden wieder aufzunehmen“. In dieser Phase, während der Wiederaufnahmeverzögerung, stellt unser Gehirn den Kontext der ursprünglichen Aufgabe wieder her. Unser Arbeitsgedächtnis konstruiert den letzten Stand der ursprünglichen Aufgabe so gut es geht. Diese Rekonstruktion kann fehlerhaft und unvollständig sein, denn das Gehirn erinnert sich in der Regel nicht an jedes einzelne Detail.

Aus diesem Grund sollten Sie es sich zur Gewohnheit machen, bei einer Unterbrechung immer den Arbeitszustand der ursprünglichen Aufgabe zu sichern. Halten Sie fest, an welcher Teilaufgabe Sie gerade gearbeitet haben, welcher Schritt unterbrochen wurde, und was als Erstes bei der Wiederaufnahme der Tätigkeit getan werden muss.

5 Arbeitsplatzforschung

Eine Forschungsarbeit bei Microsoft fand einige interessante statistische Daten zu Unterbrechungen. Computerarbeiter wurden beobachtet, interviewt und über ihre Tätigkeiten Protokoll geführt. Sie beschreiben zum Beispiel, dass jede ihrer Tätigkeiten von 0,7 Unterbrechungen gestört wird und dass die meisten Aufgaben, die sie erledigen „hohe Priorität“ haben.

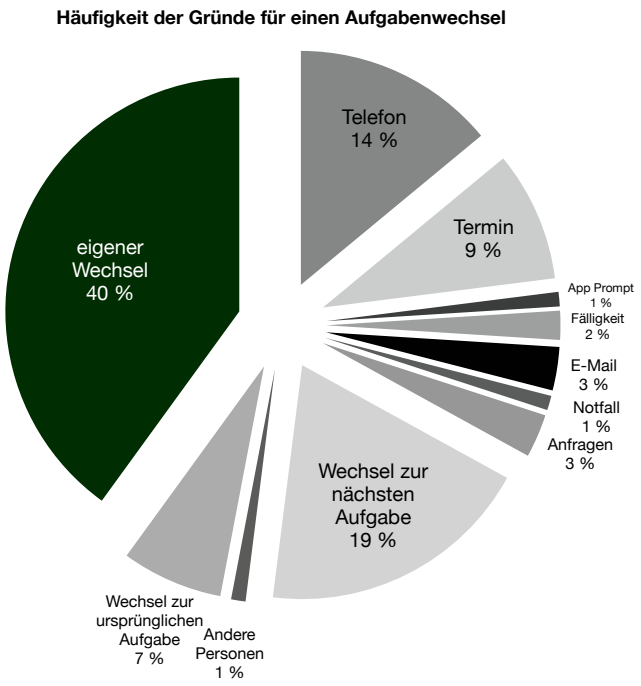


Figure 2: Gründe für Aufgabenwechsel nach Czerwinski et al.

Zur weiteren statistischen Auswertung wurde von den Forschern Videomaterial von den Computerarbeitsplätzen analysiert:

- Durchschnittlich dauert die Erledigung einer Aufgabe 53 Minuten.
- Die meisten Aufgaben waren kürzer als der Durchschnitt.
- 40 % der Wechsel zwischen den Aufgaben waren selbst ausgelöst.
- Ein Anteil von 19 % ergab sich aus dem Wechsel zur nächsten Aufgabe auf der To-Do-Liste.

Das „Zurückkehren“ zu einer Aufgabe wird von den Probanden durchschnittlich als schwieriger empfunden als der normale Aufgabenwechsel. Umfangreichere Aufgaben, die naturgemäß häufiger unterbrochen werden, dauern im Durchschnitt länger, hatten mehr Computerdokumente und waren schwieriger

wieder aufzunehmen. Viele Probanden nutzen bei Unterbrechungen Erinnerungsstützen. Als Technik wurde häufig eine an sich selbst geschickte E-Mail genutzt.

6 Tipps für die Praxis

Je nach Arbeitsplatz, Arbeitssituation und persönlichen Präferenzen bieten sich unterschiedliche Strategien an, um Unterbrechungen entweder zu vermeiden oder die negativen Auswirkungen von Unterbrechungen gering zu halten. Nachfolgend werden einige Methoden und Techniken angeboten, aus denen die passenden ausgewählt und an die eigene Arbeitssituation angepasst werden können.

R. Key Dismukes, ein Wissenschaftler des NASA Ames Research Centers, hat in einer Untersuchung einige Tipps zur Fehlervermeidung ermittelt. Auch wenn sich der eine oder andere Tipp bekannt anhört, lohnt es sich, auf die Feinheiten zu achten.

6.1 Erinnerungsstützen nutzen

Viele Menschen nutzen diese Methode bereits instinktiv. Falls Sie es noch nicht tun oder zu selten: gewöhnen Sie sich an, sich selbst Erinnerungen zu geben. Der Alarm, der Timer, die E-Mail, die Sie sich selbst senden. All diese Erinnerungen entlasten Ihr Gehirn und sorgen für weniger Fehler. Omas Knoten im Taschentuch lässt grüßen.

6.2 Multitasking vermeiden

Wie bereits erwähnt ist Multitasking nur unter bestimmten Bedingungen sinnvoll. Aufgaben, die viel bewusste Arbeitskapazität unseres Gehirns benötigen, sollten exklusiv ausgeführt werden.

Wenn eine Unterbrechung auftritt, während wir im Multitasking arbeiten, ist unser Sicherungsaufwand weit höher und damit fehleranfälliger, als wenn wir nur eine einzelne Aufgabe ausgeführen.

Mailing-Listen überfliegen und gleichzeitig die Monitoring-Ampeln kontrollieren wäre also in Ordnung. Das Einspielen eines Software-Patches mit Tests sollte besser exklusiv geschehen.

6.3 Assoziationsimpulse nutzen

Nutzen Sie physische Erinnerungshinweise. Ein physischer Erinnerungshinweis ist zum Beispiel die auf die Tischkante gelegte Bandkassette für die Datensicherung. Die Bandkassette erinnert Sie beim Verlassen des Server-Raums, dass die Bänder der Datensicherung getauscht werden müssen. Ihr Gehirn sieht die Kassette und stellt automatisch die Assoziation zur Datensicherung her. Ihnen fällt ein, was zu tun ist.

Diese Methode ist einfach realisierbar und nur durch Ihre Phantasie begrenzt. Wie wäre es, wenn Sie morgens beim Verlassen der Wohnung Ihre Sportsachen in den Flur stellen und Ihr Gehirn beim nach Hause Kommen assoziieren lassen?

6.4 Die „Nächste Aktion“ statt „To-do“

Zur erfolgreichen Eindämmung der negativen Folgen von Störungen sollten Sie immer versuchen, die aktuelle Teilaufgabe, die Sie gerade bearbeiten, noch zu Ende zu bringen. Für Ihre Arbeitsplanung bedeutet das, dass Sie Ihre Aufgaben möglichst kleinschrittig planen sollten. Durch kleinschrittige Planung verschaffen Sie sich zwei Vorteile:

1. Durch die kleinen Aufgaben vermindern Sie die Eintrittswahrscheinlichkeit einer Störung während der Aufgabenausführung. Es liegt auf der Hand, dass die Gefahr einer Unterbrechung bei lang andauernden Tätigkeiten größer ist als bei kurzen Tätigkeiten.

2. Bei einer Unterbrechung haben sie es durch die kleine Aufgabe einfacher diese noch bis zum Ende fortzuführen und abzuschliessen. Es ist bis zu einem Abschluss einfach weniger zu tun.

6.4.1 Ein Praxis-Beispiel

Formulieren Sie auf Ihrer To-Do-Liste nicht die Aufgabe „Mailserver konfigurieren“, sondern teilen Sie diese Tätigkeit etwa in folgende kleinere Nächste Aktionen auf:

- Hostnamen festlegen
- virtuelle Domainnamen setzen
- Empfängerfilter konfigurieren
- Empfängerfilter testen
- Senderfilter konfigurieren
- Senderfilter testen
- usw.

Auch wenn es sehr aufwändig erscheint die Tätigkeit durch diese vielen kleinen Arbeitsschritte zu beschreiben. Die Vorteile im Arbeitsalltag zahlen sich langfristig aus, zumal wir Computerarbeiter häufig gleiche oder ähnliche Tätigkeiten durchführen und so entweder automatisieren oder mit Vorlagen arbeiten können.

6.5 Verbesserte Checklisten benutzen

Checklisten sind seit Jahrzehnten probate Mittel, um Arbeitsabläufe zu unterstützen. Die Punkte einer konventionellen Checkliste können zwei Zustände einnehmen:

- nicht erledigt
- erledigt

Es geht aber besser. Nutzen Sie statt zwei, vier mögliche Zustände für die Aufgaben auf Ihrer Checkliste:

- nicht erledigt
- erledigt
- übersprungen
- hier fortfahren

Nutzen Sie als Kennzeichnung der verschiedenen Zustände folgende Zeichen:

- An erledigten Aufgaben wird der berühmte Haken gesetzt.
- Wenn eine Aufgabe übersprungen wird, wird Sie mit einem Kreis markiert um deutlich zu machen, dass sie nicht etwa vergessen, sondern absichtlich nicht ausgeführt wurde.
- Die Aufgabe, mit der als nächstes nach einer Unterbrechung fortgefahren wird, wird mit einem Pfeil markiert.

Diese Art die Checkliste zu führen unterstützt Sie bei der Wiederaufnahme der unterbrochenen Tätigkeit besser als konventionelle Checklisten.

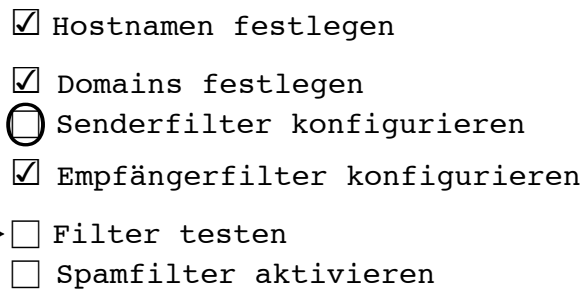


Figure 3: Verbesserte Checkliste

6.6 Timebox und Inbox nutzen

Timeboxing und das Inbox-Verfahren können helfen auftretende Störreize abzuwehren. Die Timebox stellt eine Art störungsfreie Zeitinsel dar. Mit der Timebox legen Sie eine Zeitspanne fest, in der Sie sich nur und ausschließlich um Ihre aktuelle Aufgabe kümmern. Eine Timebox, die je nach Arbeitssituation 5, 10 oder 25 Minuten dauern kann, macht Sie unantastbar, denn während die Timebox läuft, lassen Sie keine Unterbrechungen zu.

Eintreffende Ereignisse, die stören könnten, werden in Inboxen angesammelt. Diese Inboxen können sein:

- Ihre E-Mail Inbox
- Die physische Inbox an Ihrem Arbeitsplatz in der Sie selber oder andere Notizzettel und andere Dokumente platzieren können
- die Voicebox Ihres Telefons

Mit Ablauf der Timebox, bringen Sie Ihre Teilaufgabe zum Abschluss oder zu einem gesicherten Zwischenstand und wenden sich Ihren Inboxen zu. Wenn nötig, beantworten Sie E-Mails, rufen zurück und kümmern sich um angefallene Anfragen. Wenn die Inboxen zeitnah abgearbeitet werden und die Kommunikationspartner dadurch lernen, dass die Anfragen zuverlässig bearbeitet werden und die Antworten zeitnah kommen, dann wird das Timeboxing akzeptiert werden.

Allerdings funktioniert Timeboxing nicht im Alleingang. Diese Methode muss organisatorisch und kulturell vom Unternehmen, den Vorgesetzten und den Kollegen getragen werden. Nur dann kann sie ihr volles Potenzial ausspielen.

7 Fallbeispiel

Abschließend ein kleines Beispiel, wie ein Administrator die Sicherung des Zustands einer unterbrochenen Aufgabe sinnvoll durchführen könnte.

Beim Konfigurieren eines Dienstes arbeitet man in der Regel an einer Kopie der Konfigurationsdatei auf einem Testsystem. Man führt die Änderungen durch, testet und wenn alles korrekt ist, transferiert man die neue Version in das Produktivsystem. Dabei kann man natürlich unterbrochen werden. Statt nun Notizen auf Zettelchen zu schreiben, könnten Sie folgende Technik einsetzen.

Nutzen Sie die Konfigurationsdatei selbst für Ihren Erinnerungshinweis. Schreiben Sie in die Konfigurationsdatei den nächsten Schritt, den Sie erledigen müssen, wenn Sie die Aufgabe wieder aufnehmen. Schreiben Sie diesen Hinweis aber *nicht* als Kommentar in die Datei:

```
# TODO: change server name
#
# Based upon the NCSA server configuration files
```

```
#
...
```

Denn dieses Vorgehen hat den Nachteil, dass es still und leise ist. Weil die Konfigurationsdatei ohne weiteres vom Server genutzt werden kann, fällt nicht auf, dass Sie noch Änderungen durchgeführen müssen. Erst wenn Sie sich daran erinnern, dass Sie noch etwas zu tun hatten, dann in die Datei hineinschauen und den Kommentar lesen, erst dann erinnern Sie sich, dass Sie den Servernamen noch ändern müssen.

Das geht besser. Sichern Sie den Unterbrechungszustand der Tätigkeit auf eine andere, auf eine laute Art und Weise. Schreiben Sie in die Konfigurationsdatei den TODO-Hinweis *ohne* Kommentarzeichen hinein:

```
TODO: change servername
#
# Based upon the NCSA server configuration files
#
...
```

Was wird passieren? Natürlich wird es krachen! Der Server gibt beim Einlesen der Konfigurationsdatei eine Fehlermeldung aus. Denn er weigert sich das TODO als gültigen Konfigurationsparameter zu interpretieren:

```
ksbiz2:~# apache2ctl configtest
Syntax error on line 1 of /etc/apache2/apache2.conf:
Invalid command 'TODO:', perhaps misspelled or defined by a module
not included in the server configuration
```

Und genau so sollten Sie Ihre Erinnerungshinweise setzen. Nicht als stillen Kommentar, der nur dann hilft, wenn er gelesen wird. Sondern als syntaktisch fehlerhafte Eintragung. So können Sie diese Tätigkeit nicht mehr vergessen, denn der Parser passt für Sie auf und wird laut.

Gewöhnen Sie sich bei Arbeiten an Konfigurationsdateien, Scriptdateien und Quelltexten an, Ihren Unterbrechungshinweis so zu setzen, dass er so früh und so laut wie möglich auffällt. Lassen Sie Parser, Interpreter und Compiler meckern! Machen Sie es explizit statt implizit.

References

[1] Adamczyk, Piotr D., Bailey, Brian P.: *If not now, when?: The effects of interruption at different moments within task execution*, University of Illinois at Urbana-Champaign, 2004

[2] Altmann, Erik: *Brief interruptions spawn errors*, http://msutoday.msu.edu/news/2013/brief-interruptions-spawn-errors/, Michigan State University, last visited 2013-01-31

[3] Baethge, Anja, Rigotti, Thomas: *Arbeitsunterbrechungen und Multitasking*, Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, 2010

[4] Andrea Lohmann-Haislah: *Stressreport Deutschland 2012*, Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, 2013

[5] Czerwinski, Mary, Horvitz, Eric, Wilhite, Susan: *A diary study of task switching and interruptions*, Microsoft Research, 2004

[6] Dismukes, R. K.: *Prospective memory in workplace and everyday situations*, NASA Ames Research Center, 2012

[7] Palmer, Everett, Degani, Asaf: *Electronic checklists: Evaluation of two levels of automation*, NASA Ames Research Center, 1991

[8] Hunt, Andy: *Pragmatisches Denken und Lernen*, Hanser Verlag, ISBN 978-3-446-41643-7, 2009

[9] Schulz, Karsten: *Timeboxing*, http://www.lifehacker-methoden.de/lifehacker-methoden/timeboxing, Lifehacker Methoden, last visited 2013-01-31

llumos, SmartOS, OpenIndiana: Morgenröte für OpenSolaris oder Sonnenfinsternis?

Volker A. Brandt
Brandt & Brandt Computer GmbH
Am Wiesenpfad 6
53340 Meckenheim
Deutschland
<vab@bb-c.de>

1 Zusammenfassung

Nach der Übernahme von Sun durch Oracle wurde die OpenSolaris-Distribution nicht mehr weiter gepflegt, und die als Open Source durchgeführte Entwicklung findet wieder hinter verschlossenen Türen statt.

In diese Lücke stieß zunächst *Illumos*. Ziel der Initiatioren von Illumos ist eine freie und vollständige Version einer Solaris-kompatiblen Systemumgebung. Bald entwickelten sich diverse Distributionen um diesen Kern: OpenIndiana, SmartOS, Illumian, OmniOS, StormOS, DilOS und noch diverse weitere. Und natürlich gibt es Solaris auch noch bei Oracle selbst.

Dabei reicht die Palette vom Ein-Mann-Hack (Tribblix) bis zum kommerziell verwendeten Cloud-Provisionierungs-Tool (SmartOS). Hier soll eine Auswahl dieser Distributionen aufgelistet und exemplarisch kurz vorgestellt werden.

Der im Folgenden verwendete Begriff „x86" meint alle 32- und 64-Bit-Architekturen mit Intel- und AMD-CPUs.

2 Freigabe von großen Teilen des Solaris-Quellcode als Open Source

Die Freigabe von fast dem gesamten Solaris-Quellcode als Open Source war ein wichtiger Schritt in der langen Geschichte dieses Flaggschiff-Betriebssystems des Unix-Pioniers Sun Microsystems. Als erste Komponente des Systems wurde das inzwischen auch auf einige andere Plattformen portierte *DTrace* veröffentlich, gefolgt von fast allen anderen Systembestandteilen. Ausgenommen blieben lediglich diejenigen Codes, für die Sun Microsystems nicht über die notwendigen Rechte verfügte, weil diese Systembestandteile von Dritten erstellt wurden (meist Gerätetreiber, aber auch einige andere Komponenten).

3 OpenSolaris-basierte Distributionen

Zunächst gab es von Sun selbst zwei Distributionen von Solaris. Zum einen war da das Solaris 10, das als kommerziell supportete stabile Variante des OS das Brot-und-Butter-Geschäft darstellte, zum anderen bekam man aber auch die Vorab-Entwicklungsversion der kommenden Version Solaris 11, diese unter dem Codenamen „Nevada". Bemerkenswert dabei war insbesondere die Tatsache, daß Solaris 10 auch für kommerzielle Zwecke eingesetzt werden durfte, was einen deutlichen Wandel zu den bis Solaris 9 geltenden Nutzungsbedingungen darstellte. Auch Nevada durfte frei verwendet werden. Mitte 2008 wurde eine auf Nevada basierende „freie" Solaris-Version veröffentlicht: *OpenSolaris*[9].

Die Verfügbarkeit des Qellcodes eröffnete auch für andere die Möglichkeit, auf Solaris basierende Distributionen anzubieten. So erschienen diverse Distributionen auf OpenSolaris-Basis, die sich in Umfang und Zielrichtung unterschieden, beispielsweise Schillix (mit System-V-Paketen statt dem *Image Packaging System*-Format (IPS) von OpenSolaris) oder Belenix (mit KDE statt Gnome als Desktop) und einige mehr.

Die Firma Nexenta[5] verwendete OpenSolaris und insbesondere das darin enthaltene *Zettabyte File System* (ZFS), um leistungsfähige und günstige Storage-Systeme zu vermarkten. Ihre Solaris-Version stellte Nexenta unter dem Namen *NexentaOS* und die Storage-Software als *NexentaStor*[4] im Binärformat zur kostenlosen, aber eingeschränkten Nutzung zur Verfügung.

4 Der neue Hausherr: Oracle

Nach der Übernahme von Sun Microsystems durch Oracle Mitte 2010 änderte sich das Bild dramatisch: Die Nutzungsbedingungen für Solaris wurden ab der Version 10 U9 wieder verschärft; die Version Solaris 10 U8 ist die letzte, die für beliebige Einsatzzwecke legal verwendet werden darf. Alle weiteren Versionen auch von Solaris 11 dürfen nur noch für Entwicklungszwecke kostenfrei eingesetzt werden.

Die OpenSolaris-Releases wurden nicht mehr aktualisiert, und die Veröffentlichung der Commits im Source-Repository von Sun wurden nicht mehr zum externen Zugriff bereitgestellt (mit Ausnahme einiger weniger Komponenten wie der IPS-Software). Dies wurde unter anderen damit begründet, daß Dritte mit dem geistigen Eigentum der Firma Oracle Profit erzielen, ohne Oracle daran teilhaben zu lassen. Gemeint war natürlich die Firma Nexenta.

Nach der Übernahme und im Verlauf der Jahre 2010 und 2011 verließen immer mehr Sun-Mitarbeiter die Firma Oracle. So arbeiten heute alle ursprünglichen Entwickler sowohl von ZFS als auch von DTrace nicht mehr bei Oracle.

5 Die Basis: Illumos

Zunächst hatte Oracle noch in Aussicht gestellt, daß nach jedem „vollen Release" einer neuen Solaris-Version auch der Source Code allgemein freigegeben werden sollte. Ziemlich schnell wurde aber klar, daös Oracle keineswegs die Absicht hatte, dies zu tun.

Sowohl die Systemdistribution als auch den Source Code hatte Sun seinerzeit OpenSolaris genannt. Letzterer war in verschiedenen öffentlichen Repositories, *Gates* genannt, organisiert. Der „Kern" des Systems bestand hauptsächlich aus den Code-Teilen für den Kernel, Nerzwerk-Unterstützung und das Userland. Kernel- und Netzwerk-Source wurden unter dem Begriff *OS and Network* (ON) zusammengefaßt.

Die letzte verfügbare Source-Version von ON entsprach dem Build 151. Diese war aber nicht ausreichend, um eine freie Systemversion zu bauen – es fehlten noch einige „Kleinigkeiten", die bisher immer von Sun als Binärdateien mitgeliefert worden waren. Diese Lücke wollte Garrett D'Amore, ein ehemaliger Sun-Mitarbeiter und seinerzeit bei Nexenta beschäftigt, schließen. Er gab den Anstoß, die noch fehlenden Teile neu zu implementieren und daraus eine freie, eigenständig verwendbare Distribution der ON-Komponenten zu schaffen: *Illumos*[1]

Illumos (das eigentlich immer klein „illumos" geschrieben werden soll) ist also keine eigene Distribution, sondern stellt die Basis für alle anderen modernen von OpenSolaris abgeleiteten Distributionen dar. Ohne Illumos gäbe es all die unten aufgeführten Systemvarianten sehr wahrscheinlich nicht.

6 Die Distributionen

6.1 OpenIndiana

Nachdem mit Illumos der Grundstein für ein von Oracle autarkes System gelegt worden war, lag der Gedanke nahe, den Rest von OpenSolaris, also das Userland (das teilweise noch upstream bei Oracle verfügbar war und ist), den Desktop (eine angepaßte Variante von Gnome2) sowie die übrigen Komponenten nachzubauen. Diese Arbeit wurde von Alasdair Lumsden, einem britischen Hosting-Anbieter und Solaris-Enthusiasten, in Angriff genommen. Er nannte sein Projekt *OpenIndiana* (OI) und leitete es auch bis Mitte 2012.

OpenIndiana startete sehr ambitioniert: Es sollte ein universelles Betriebssystem für Server und Desktops sein, sollte die noch verfügbaren Oracle-Sources tracken, sollte möglichst aktuelle Versionen der FOSS-Komponenten im OpenSolaris liefern, möglichst binärkompatibel zu den Oracle-Distributionen und auch noch einfach zu installieren sein. Dank dem hohen Einsatz einzelner Personen wurde auch eine brauchbare Version gebaut und der Allgemeinheit zur Verfügung gestellt.

Mit der Laufzeit des Projektes stellten sich aber auch die ersten Risse ein: Die Belastung der einzelnen Personen wurde immer größer, und es gab auch Dissonanzen mit den Illumos-Kollegen. Irgendwann warf Lumsden das Handtuch, seitdem gibt es keine offizielle Projektleitung mehr.

Obwohl Illumos zunächst x86- und SPARC-Architekturen gleichberechtigt unterstützen wollte, fokussieren sich die Sponsoren der Entwickler inzwischen ausschließlich auf x86-Hardware. Deshalb gibt es keine offizelle SPARC-Version von OI, obwohl außer mangender Unterstützung aktueller Grafik-Hardware keine technischen Gründe dagegensprechen.

6.2 NexentaOS/Illumian und StormOS

Die Firma Nexenta hatte zunächst den größten Vorsprung bei den Solaris-basierten Distributionen. NexentaOS war anfangs direkt auf dem OpenSolaris-Kernel aufgebaut, verwendete aber ein Ubuntu-Userland; eine graphische Benutzeroberfläche stellte Management-Funktionen für die von Nexenta angebotenen Storage-Server bereit.

Nachdem die Basis auf Illumos umgestellt wurde, änderte man auch den Namen von NexentaOS in *Illumian*, um die Symbiose aus einem Illumos-basierten System mit Debian-Paketverwaltung anzudeuten. Ein zunächst von einem einzelnen Entwickler angestoßenes Projekt zur Erstellung einer Debian-basierten Desktop-Variante von NexentaOS mit dem Window-Manager xfce namens *StormOS* ist zwischen wieder eingeschlafen. Nexenta wird immer wieder vorgeworfen, daß sie nicht zwar die Arbeit anderer in ihren kommerziellen Angeboten verwenden, nicht aber ihre eigenen Entwicklungen anderen zugänglich machen würden. So kommt es immer wieder zu erregten Diskussionen mit Entwicklern von anderen Illumos-basierten Distributionen.

6.3 Oracle Solaris 11

Auch Oracle selbst hat natürlich weiterhin Distributionen im Angebot, die in der Nachfolge von OpenSolaris stehen. Die aktuellste Version ist Solaris 11.1, das zwar offiziell als „Cloud"-Betriebssystem vermarktet wird, aber durchaus als vollwertiges Desktop-System verwendet werden kann. Die x86- und SPARC-Architekturen sind gleichwertig, allerdings gibt es ein Live-Image nur in der x86-Variante.

Die Nutzungsbedingungen von Solaris 11.1 sind aber radikal anders als jene für OpenSolaris, und die Entwicklung wird nur noch von Oracle-Mitarbeitern hinter verschlossenen Türen durchgeführt. Daher eignet es sich eigentlich nur für kommerzielle Einsätze mit vorhandener Solaris-Lizenz und gegebenenfalls bei Oracle eingekauften Support.

6.4 SmartOS

Eine der großen Anwender von Solaris für Virtualisierung und Cloud-Provisionierung der ersten Stunde war die Firma Joyent[2]. Als Oracle Sun übernahm und sich die Rahmenbedingungen für Anbieter von Solaris-basierenden Lösungen gravierend änderten, sah man die Chance, sich mit Illumos von Oracle zu emanzipieren. Das heutige Joyent-Produkt *SmartOS* ist sehr auf virtualisierte Server zugeschnitten; eine Desktop-Unterstützung oder SPARC-Versionen sind unnötig. Joyent ist heute nach Oracle der größte Player im Solaris-Markt; viele hochkarätige Solaris-Entwickler arbeiten heute dort. Joyent ist Hauptsponsor von Illumos und in geringerem Maße auch von OpenIndiana, und hat mehrere qualitativ neue Entwicklungen für Solaris durchgeführt, insbesondere einige ZFS-Innovationen und den Port des von Linux stammenden *Kernel-based Virtual Machine* (KVM)[3], auch für AMD-Prozessoren.

6.5 OmniOS

Die *OmniOS*[6]-Distribution entstand aus der Notwendigkeit der Firma OmniTI[7], eine möglichst ressourcenschonende OS-Version für Hosting- und Virtualisierungsanwendungen zu haben. Nachdem OmniOS auf Illumos basiert, entschloß man sich, es frei und kostenlos zur Verfügung zu stellen. OmniOS ist mit SmartOS vergleichbar, aber verfolgt einen deutlich minimalistischeren Ansatz als letzteres.

6.6 Tribblix

Stellvertretend für diverse Ein-Mann-Projekte steht *Tribblix*[13]. Peter Tribble ist seit vielen Jahren in der Solaris-Szene aktiv und beschäftigt sich mit Systemadministration und Systemmanagement. Er wünschte sich ein Desktop-System mit einem minimalen Window-Manager und begann einfach auf Illumos-Basis mit der Entwicklung. Er beschreibt offen seinen Entwicklungsprozeß und läßt jeden daran teilhaben.

6.7 OpenSXCE

Das andere Ende des Spektrums der Ein-Mann-Projekte stellt *OpenSXCE*[10] dar, das es nur als SPARC-Live-Image gibt. Es wurde von Martin Bochnig als OpenIndiana-SPARC-Version zusammengestellt; es soll als Proof-of-Concept demonstrieren, daß auch heute noch ein sinnvolles Desktop-Betriebssystem auf einer SPARC-Plattform betrieben werden kann. Das DVD-Image ist ebenfalls frei verfügbar.

7 Ausblick

Wie man sieht, ist die Bandbreite der OpenSolaris-Nachfolger gar nicht so klein. Aber welche Distribution soll man denn nun nehmen? Welche ist „zukunftssicher"? Gibt es denn **den** OpenSolaris-Nachfolger?

Die Antwort: Ein klares „jein". Vom Ansatz und vom Anspruch her ist sicher OpenIndiana der Nachfolger. Die (teilweise durch mangelnde Ressourcen und die Ausrichtung des darunterliegenden Illumos verursachten) Beschränkung auf x86-Plattformen schränkt dies aber erheblich ein, insbesondere wenn man ältere und vielleicht nicht mehr von Solaris 11.1 unterstützte Hardware betreiben will.

Vor der Entscheidung für eine spezielle Distribution muß mehr denn je bedacht werden, wofür das Zielsystem eigentlich eingesetzt werden soll. Will man Virtualisierung, sind SmartOS und OmniOS die erste Wahl. Soll es ein Desktop sein, und man scheut sich nicht, selbst Hand anzulegen, macht Tribblix oder eine System-V-Paket-basierte Distro wie *Schillix*[11] sicher mehr Spaß. Ist SPARC das Ziel, nimmt man OpenSXCE, wenn es ausgereift ist.

Zum Einsatz von ZFS braucht man vielleicht gar kein Solaris, sondern kann auch FreeBSD oder ein darauf basierendes NAS-System verwenden. Und will man nur Solaris-Applikationen betreiben, gibt ein Einsatz von Solaris 10 U8 vielleicht mehr Sinn als alle neueren Systeme.

In einer idealen Welt hätten wir ein OpenIndiana auf Illumos-Basis mit vielen Treibern für moderne Hardware, gleichberechtigtem x86- und SPARC-Support, einem aktuellen Desktop mit Wahlmöglichkeit zwischen Gnome, xfce und KDE, und den neuesten Solaris-Kernel-Features in ZFS, DTrace und Crossbow aus SmartOS. Dann würde doch wieder die Sonne auf die Solaris-Anhänger scheinen.

Man wird doch wohl noch träumen dürfen...

Literatur

[1] http://www.illumos.org/

[2] http://www.joyent.com/

[3] http://www.redhat.com/resourcelibrary/whitepapers/doc-kvm

[4] http://www.nexentastor.org/

[5] http://www.nexenta.com/corp/

[6] http://omnios.omniti.com/

[7] http://www.omniti.com/

[8] http://www.openindiana.org/

[9] http://www.opensolaris.org/

[10] http://www.opensxce.org/

[11] http://schillix.berlios.de/

[12] http://smartos.org/

[13] http://www.tribblix.org/

IPv6-Marketing kritisch hinterfragt

Marc Haber
incluesion
Schriesheimer Str. 8
68549 Ilvesheim
deutschland
<mh+ffg2013@zugschlus.de>

1 Abstract

Dass IPv6 kommen wird, ist seit fünfzehn Jahren keine Frage mehr. Die Frage ist eher, wann es kommt, und wie es kommt. Während viele IPv6-Anwender und -Fans das klassische IPv4 schon als "legacy IP" bezeichnen und die Aktivierung von IPv6 gerne als "ganz einfach, geht alles automatisch" beschreiben, geht es in diesem Artikel schwerpunktmäßig um die Überraschungen, die den Systemadministrator bei der Aktivierung von IPv6 erwarten.

Der Vortrag wird eine Reihe der üblichen Aussagen über IPv6 in der Praxis kritisch hinterfragen und erläutern, warum es eben doch nicht so einfach ist.

2 Einführung

IPv6 kommt innerhalb der nächsten zwei Jahre. So heißt es seit den späten 1990er Jahren. Bisher ist diese Voraussage im großen Stil noch nicht eingetreten und wir bewegen uns eher in kleinen als großen Schritten, aber die Dinge bewegen sich. Im Rahmen des zweiten IPv6-Days im Juni 2012 haben viele Unternehmen IPv6 eingeschaltet und es im Gegensatz zum ersten IPv6-Day 2011 am Ende des Tages nicht wieder abgeschaltet. Bei etlichen Hostern bekommen neue Webpräsenzen automatisch IPv6 zusätzlich zum IPv4, auf so mancher Internetanbindung für Privatkunden findet man IPv6 in mehr oder weniger verdeckter Form und man wird von Hardware-Lieferanten nicht mehr ausgelacht, wenn man nach IPv6 fragt. Im Gegenteil: Das Lachen ist durch betretenes Schweigen ersetzt.

Inzwischen fangen selbst kleine und mittlere Unternehmen an, IPv6 für ihre externen und internen Dienste zu verwenden, wenn auch allerdings oft aus der Not: Es ist in den letzten Jahren zunehmend schwieriger geworden, neue IPv4-Adressen zugewiesen zu bekommen; neue providerunabhängige IPv4-Adressen bekommt man gar nicht mehr.

Viele Veröffentlichungen über IPv6 loben das neue Internetprotokoll in den höchsten Tönen und erklären in epischer Breite, was mit IPv6 alles einfacher wird als mit “legacy IP”, wie IPv4 in den Kreisen derer genannt wird, die IPv6 einsetzen, weil sie Spaß an der Neuerung haben.

Leider ist auch IPv6 nicht das Allheilmittel. IPv6 räumt mit vielem auf, was in der Technikergemeinde als Altlast gesehen wird und in der Tat sind viele Dinge, die uns bei IPv4 jeden Tag in unserer Effizienz behindern oder uns gar zu vermeidbaren Fehlern verleiten, in IPv6 nicht mehr vorhanden. Aber es gibt auch viele Dinge, die von den Protokolldesignern als unnötig eingestuft wurden, den Administratoren aber so wichtig waren, dass die Einführung von IPv6 durch ihre Abwesenheit verzögert wurde.

3 IPv6

Über IPv6 selbst braucht man glücklicherweise nicht mehr so viele Worte zu verlieren. Eine kurze Vorstellung soll es trotzdem sein.

3.1 Adressierung

Wir haben Adressen mit einer Länge von 128 Bit, was uns erlaubt, die wie bei IPv4 vorliegende Trennung zwischen Netzwerk- und Host-Teil der Adresse fest auf die Mitte der Adresse zu legen. Das mag auf den ersten Blick wie eine schreckliche Verschwendung vorkommen, aber wenn man es sich näher betrachtet, sind 64 Bit für die Auswahl des Netzes genug Adressen auch für den Fall, dass die interstellare Raumfahrt und die Kommunikation mit Überlichtgeschwindigkeit möglich wird und es somit sinnvoll sein kann, die Internets mehrerer Planeten miteinander zu verbinden.

Auf der anderen Seite sind die 64 Bit, die für den Host-Teil der Adresse vorgesehen sind, großzügig genug, um automatische Adressvergabe ohne eine zentrale Instanz zu ermöglichen. Das weit verbreitete Verfahren der *Stateless Autoconfiguration* beispielsweise nimmt die – laut Spezifikation weltweit eindeutige – MAC-Adresse des Ethernet-Interfaces des Hosts, transformiert sie nach einem standardisierten Verfahren und erhält somit den Host-Teil der IPv6-Adresse des Hosts.

Die fehlerträchtige und zu verwirrenden Sonderkonstruktionen verleitende Notation mit der Netzmaske wird aufgegeben. In IPv6 gibt es nur noch die Prefixnotation, in der die nach einem Schrägstrich aufgeschriebene Dezimalzahl anzeigt, an welcher Stelle der binär geschriebenen Adresse der Netzwerkteil endet. Diese Notation ist auch schon in IPv4 üblich; viele Software unterstützt aber nach wie vor nur die Netzmaskennotation.

Im derzeit vergebenen Teil des IPv6-Adressraums ist es üblich, dass die ersten 32 Bit der Adressen den Provider identifizieren, die nächsten 16 Bit der Adressen den Kunden des Providers, und jeder Kunde dann noch einmal 16 Bit zur Verfügung hat, um seinen Adressraum in Subnetze zu unterteilen, ohne den Host-Teil seiner Adressen kürzer machen zu müssen und damit auf die Stateless Autoconfiguration verzichten zu müssen. Dabei werden die /32-Netze der Provider nicht aufsteigend vergeben, sondern es wird dazwischen Platz gelassen, um einem Provider zu ermöglichen, seinen Adressraum zu vergrößern ohne mit mehreren Netzen operieren zu müssen. So soll es der Regelfall werden, dass ein aus seinem /32-Netz herausgewachsender Provider einfach das dazu passende zweite /32-Netz hinzu bekommt und aus den beiden /32-Netzen dann einfach ein /31 macht. Das Wachstum der weltweiten Routingtabellen soll auf diese Weise in beherrschbarem Rahmen gehalten werden.

Ein hauptsächlich für Privatkunden operierender Provider wird die Grenze zwischen Providernetz und Kundennetz von /48 auf /56 oder gar /60 verschieben, um sich zu erlauben, mehr als 16 Bit für die Adressierung seiner Kunden zu verwenden und dafür in Kauf nehmen, dass jeder seiner Kunden nur noch 256 oder gar nur noch 16 Subnetze bilden kann.

3.2 Mehrere Adressen pro Host

Viele IPv6-Mechanismen basieren darauf, dass ein Host mehrere IP-Adressen haben kann und im Regelfall auch hat. Dies ermöglicht die einfache Trennung verschiedener Dienste und unterstützt Migrationsszenarien.

Eine Adresse kann eine Lebensdauer haben, nach deren Ablauf sie nicht mehr für neue ausgehende Verbindungen verwendet wird, der Host aber noch auf dieser Adresse erreichbar ist (*deprecated*). Auf diese Weise können Verbindungen zur alten Adresse aufrecht erhalten oder kontrolliert abgebaut werden.

3.3 Autoconfiguration

Ein IPv6-Router kann im Netz bekanntgeben, dass er eine Verbindung nach “draußen” anbietet. Dabei ist auf ihm ein Prefix konfiguriert, und diesen Prefix schreibt er in seine Bekanntgaben (*Announcements*) hinein. Damit steht der Netzwerkteil aller IP-Adressen in diesem Netz bereits fest. Ein Host kann

dann nach verschiedenen Verfahren einen Host-Teil der Adresse ermitteln, diesen mit dem vom Router empfangenen Netzwerkteil verbinden und hat eine vollständige und benutzbare IP-Adresse. Ein Router, der geplant außer Betrieb geht, gibt dieses auch im Netz bekannt. Die Hosts setzen dann die Adressen aus seinem Prefix auf *deprecated* und verwenden sie nicht mehr aktiv.

Dieser Prozess funktioniert nahtlos auch, wenn mehrere Router im Netz sind. Jeder Host kann sich dann eine IP-Adresse aus jedem Prefix autokonfigurieren und ist damit in der Theorie über verschiedene Wege erreichbar.

Eine per Stateless Autoconfiguration zugewiesene IP-Adresse hängt von der MAC-Adresse des Hosts ab. Das bedeutet auch, dass sich die IP-Adresse des Hosts ändert, wenn man die Hardware austauscht. Auch enthält der ursprüngliche Standard der Stateless Autoconfiguration keine Methoden zur Bekanntgabe von DNS-Servern und weiteren Konfigurationsdaten, so dass es hier eine Vielzahl von Erweiterungen und sogar DHCP-Dialekte gibt, um den Standard von IPv4 wieder zu erreichen.

3.4 Link-Local-Adressen

Jeder IPv6-Host gibt sich auf jedem Netzwerklink, an den er angeschlossen ist, eine Link-Local-IP-Adresse. Hierfür ist der Prefix fe80::/32 vorgesehen. Dies ist notwendig, weil mit IPv6 zahlreiche Verwaltungsprotokolle nicht mehr auf Broadcast, sondern auf Multicast aufsetzen. Auch ist es sehr praktisch, sich beim Aufbau von Netzen oder im Störungsfall auch auf einem un- oder fehlkonfigurierten Link per IP von Gerät zu Gerät hangeln zu können. Da auf jedem Netzwerklink derselbe Prefix verwendet wird, sind Link-Local-Adressen nicht routebar und aus Sicht der Software auf einem Host nur dann eindeutig, wenn der Interfacename dazu geschrieben wird, so zum Beispiel `fe80::1%eth0`.

3.5 Unique-Local-Adressen

Eigentlich soll man beim Aufbau eines IPv6-Netzes Adressen verwenden, die einem über einen Provider zugewiesen wurden. Eine Alternative wie die IPv4-Adressen nach RFC1918 ist nicht vorgesehen. Da dies in der Nutzerbasis nicht besonders populär war, wurde zuerst ein sehr stark an RFC1918 angelehnter Adressbereich für “site local” Adressen angelegt, der jedoch später zugunsten von Unique-Local-Adressen wieder abgeschafft wurde. Hierbei ist ein Verfahren spezifiziert, das die Kollision von Unique-Local-Adressen zweier Organisationen unwahrscheinlich machen soll.

3.6 Mobile IPv6 und IPSEC

IPSEC war ursprünglich Bestandteil der IPv6-Definition wurde aber 2011 zu einer optionalen Erweiterung heruntergestuft. Schon vorher wurde IPSEC in vielen IPv6-Implementierungen nicht umgesetzt. Ebenfalls nicht in signifikante Verbreitung gekommen ist Mobile IPv6, mit dessen Hilfe bewegliche Netzteilnehmer eine feste IP-Adresse erhalten sollen, ohne dass der Datenverkehr Umwege nimmt.

3.7 Nicht vorgesehen

Neben den oben bereits erwähnten Netzmasken sind in IPv6 auch Dinge weggefallen, die der Netzgemeinde lieb und teuer geworden sind. Hierzu zählen beispielsweise providerunabhängige IP-Adressen, Network Address Translation und die direkt ohne weitere Maßnahmen intern benutzbaren IP-Adressen, wie sie RFC 1918 für IPv4 bietet. Viele dieser Dinge sind technisch ohne weiteres implementierbar, wurden aber in den Standards für IPv6 nicht berücksichtigt, weil man diese als Krücken angesehenen Dinge nicht mehr benötigt.

Die Anwender stellt die Entfernung dieser Dinge vor die Herausforderung, umlernen zu müssen. In vielen Organisationen ist jedoch die Bereitschaft zu so weit gehendem Umlernen nicht vorhanden, so dass beispielsweise das Fehlen des oft als Sicherheitsfeature mißverstandenen Network Address Translation oder Masquerading nicht nur einmal die Einführung von IPv6 verhindert hat. Dabei ist es oft nicht relevant gewesen, dass jetzt jeder interne Host eine offizielle, weltweit eindeutige Adresse haben kann,

da in den Security-Policies oft aus IPv4-Zeiten festgeschrieben wurde, dass interne IP-Adressen nicht nach draußen dringen dürfen.

Außerdem haben viele Organisationen die Aussage, man bräuchte keine providerunabhängigen IP-Adressen, weil das Einführen neuer IP-Adressen mit IPv6 so viel einfacher ist als mit IPv4, zu Recht nicht glauben wollen und waren nicht bereit, sich mit providerzugeordneten Adressen an einen Provider zu binden.

4 Überraschungen bei IPv6

4.1 Renumbering ist einfach

Diese Aussage steht in jeder Publikation über IPv6. Sie ist auch richtig, wenn es einem nur darum geht, Hosts per IP-Adresse anpingen zu können. In der Praxis möchte man allerdings mehr: Man möchte richtige Dienste auf der IP-Adresse betreiben, man möchte ein Grundpensum an Security betreiben können und man möchte DNS haben.

Üblicherweise wird beim Renumbering von IPv6-Systemen nur der Fall betrachtet, dass beispielsweise der Provider gewechselt werden soll, und dass deswegen ein neuer Prefix verwendet werden soll. Hat man nur autokonfigurierte Hosts im Netz, ist das wirklich einfach: Man schaltet Route Advertisements auf dem neuen Router ein, setzt die preferred lifetime auf dem alten Router auf Null und wartet. Nach einiger Zeit verwenden die Hosts nur noch die IP-Adresse aus dem neuen Prefix, die Verbindungen auf die alten IP-Adressen sind alle abgebaut, wir schalten den alten Router ab und fertig.

Haben wir nur Autoconfiguration, nur eine Adresse pro Prefix und Host verwendet, dynamischen DNS, nirgendwo Accesslisten und Software die mit Adresswechseln klar kommt, sind wir in der Tat wirklich fertig. Aber wo ist das schon so?

4.1.1 Statische IP-Adressen

Haben wir statische IP-Adressen vergeben, ist das Renumbering Handarbeit wie bei IPv4 auch. Neue Adresse hinzukonfigurieren, DNS ändern, alte Adresse wegkonfigurieren, und sich eventuell darüber wundern, das die Software, die auf dieser IP-Adresse einen Dienst erbringt, nur eingeschränkt konfigurierbar ist.

Die Steigerung des Spaßes ist übrigens, wenn man für jeden Dienst eine eigene IP-Adresse vorgesehen hat und Wert darauf legt, dass die Dienste auch nur auf diesen erreichbar sind und nicht auf allen Adressen, die der Host gerade hat.

4.1.2 Dynamischer DNS

Dynamischer DNS ist ein Thema für sich, deswegen sind die technischen Grundlagen in einem Exkurskasten beschrieben. Beim dynamischen DNS trägt sich ein Host mit seinen IP-Adressen selbst in den DNS ein, indem er seinen Namen und seine IP-Adresse an den DNS-Server gibt. Hat man Dynamischen DNS, wir das Renumbering von IPv4 und IPv6 deutlich einfacher. In der Migration möchte man allerdings kontrollieren können, ob der Hostname auf alle oder nur auf eine der IP-Adressen zeigen soll und auf welche.

4.1.3 Accesslisten

Oft hat man IP-Adressen von Hosts auf Netzwerkkomponenten, Firewalls oder anderen Hosts in Accesslisten stehen und legt auf diese Weise fest, welcher Host wohin IP-Pakete schicken darf. Renumbert man einen Host, muss man alle diese Accesslisten anfassen, was besonders aufwendig ist, wenn sie von Dienstleistern, Kunden oder Partnerunternehmen gepflegt werden. Dann muss man nämlich oft begründen, warum man eine zweite Adresse eingetragen haben möchte oder warum die Änderung überhaupt notwendig ist.

4.1.4 Ungewollt automatisch IPv6 und nichts geht mehr

Ein immer wieder gerne auftretender Fehler ist, dass auf einem Netz IPv6 aktiviert wird, und nicht alle Hosts und Applikationen wirklich darauf vorbereitet sind. Dann fangen die Applikationen unter Umständen an, unkoordiniert und unerwartet IPv6 zu benutzen, was dazu führt, dass die Verbindungen plötzlich an einer im Weg befindlichen und noch nicht für diese Applikation freigeschalteten Firewall scheitern. Dies sind die Fehler, nach denen man Stunden sucht.

4.2 LAN mehrfach anbinden

Dank der Autoconfiguration soll es möglich sein, einem LAN mehrere Ausgänge ins Internet zu geben, ohne jeden Host einzeln anfassen zu müssen. Das sieht in der Theorie sehr schon aus: Der zweite Router sendet Router Advertisements, die Hosts lernen den Prefix, geben sich eine IP-Adresse in dem zweiten Netz und benutzen sie einfach.

In der Praxis wird die Adresse aller Router als Defaultroute eingetragen, und man hat im Router Advertisement nur eingeschränkt die Möglichkeit, eine Priorität vorzugeben. Jeder Host wird also für sich entscheiden, welchen der Router er als Defaultgateway verwendet - und dies auch für Pakete tun, die als Absender eine Adresse aus einem der anderen Prefixe haben. Wenn die Router voneinander wissen, kann man sie so konfigurieren, dass die Pakete mit dem “falschen” Absender an den “richtigen” Router weitergeleitet werden und somit auf dem “richtigen” Uplink herausgehen und nicht am vielleicht vorhandenen Reverse Path Filtering des ISPs zerschellen.

Wenn die Router nicht voneinander wissen, müssen beide ISPs auch Pakete routen, deren Absender aus Sicht des jeweiligen ISPs nicht aus dem Kundennetz stammt. Das sieht für den ISP wie Adreßspoofing aus, was er wegen des hohen Mißbrauchspotenzials im Default nicht erlauben wird.

Die dritte Möglicheit wäre, auf allen Hosts im LAN Policy Routing zu aktivieren und auf diese Weise sicherzustellen, dass jeder Host seinen ausgehenden Traffic zum richtigen Router schickt. Das konterkariert die Autoconfiguration komplett und macht die Migration schmerzhafter als sie mit IPv4 je war. Bei statischer Konfiguration mag dieser Weg gangbar sein - Wünschenswert ist er nicht.

4.3 Mal eben schnell - nicht mit IPv6

Grundsätzlich muss man bei IPv6 deutlich koordinierter und vorbereiteter an die Dinge herangehen, als das bei IPv4 notwendig war. Die Auswirkungen der eigenen Aktionen sind nicht so lokal, wie sie bei IPv4 mit RFC1918 und NAT wären.

Ein alltägliches Beispiel: Ein Consultant hat bei seinem Kunden ein Notebook im Netz, bekommt eine IPv4- und eine IPv6-Adresse zugewiesen und kann arbeiten. Nun möchte er auf diesem Notebook eine VM starten und diese VM ans Netz bringen. Die VM einfach auf das Netz bridgen geht nicht, weil die Sicherheitsmaßnahmen des Kunden den Switchport abschalten, wenn mehr als eine MAC-Adresse gesehen wird.

Bei IPv4 ist das einfach: Der Consultant benutzt das Betriebssystem des Notebooks als Router, gibt der VM eine RFC1918-Adresse und nattet den ausgehenden Traffic der VM auf die IPv4-Adresse des Notebooks. Fertig.

Mit IPv6 ohne NAT steht der Consultant im Regen, muss für das Netz zwischen Host und VM ein ULA-Netz konfigurieren und die notwendige Konnektivität mit Application Level oder Circuit Level Gateways herstellen - mit allen Einschränkungen, die die meisten von uns längst verdrängt haben.

Selbst wenn das LAN kooperativer ist als in dem Beispiel des Consultants beim Kunden, muss neben dem Host und der VM noch der Router angefasst werden, um ein /64 für die VMs hinter dem Host zum Host zu routen - und wenn mehrere Router im Netz sind, jeder von ihnen - es sei denn, es ist ein dynamisches Routingprotokoll wie OSPF im Einsatz. Das wird man aber in einem reinen IPv4-Netz mit weniger als einer Hand voll Routern nicht haben - IPv6 wird die Benutzung von dynamischem Routing in kleinen Netzen deutlich fördern, weil man viel häufiger zusätzliche Netze einrichten muss.

4.4 ping6 geht - was nun?

Ein funktionierendes Ping bedeutet zwar, dass unser Host grundsätzlich per IPv6 erreichbar ist, beziehungsweise selbst per IPv6 kommunizieren kann, aber das heißt noch lange nicht, dass das Projekt "IPv6" abgeschlossen werden kann. Die Arbeit geht jetzt erst richtig los. Wir wissen alle schon lange, dass es keine schlaue Idee ist, in Applikationen, Konfiguationen etc direkt IP-Adressen zu verwenden. Das tut schon mit IPv4 dann weh, wenn sich Änderungen am Netz ergeben. Mit den langen IPv6-Adressen wird das vollständig unpraktisch, besonders weil sich die Software noch nicht darauf geeinigt hat, wie man die Kombination von IP-Adresse und Port schreibt. Was bei IPv4 einfach `192.0.2.3:25` geschrieben wird, kann man in IPv6 nicht einfach `2001:db8:1::3:25` schreiben. Die in echter Software hierfür gefundenen Notationen enthalten meistens eckige Klammern oder ähnliches und sind selten abschließend dokumentiert, aber ein bisschen Experimentieren braucht es eigentlich immer. Also möchte man DNS verwenden. Hat man alle seine Hosts statisch konfiguriert, ist das auch genau so einfach wie bei IPv4. Aber halt auch nicht einfacher. Wie weiter oben und im Kasten erwähnt, kann Dynamic DNS einem hier helfen, erzeugt aber auch wieder eigenen Aufwand.

4.5 Herausgeforderte Software

Wenn wir jetzt Netz und Betriebssystem im Griff haben, kommen wir zur nächsten IPv6-Herausforderung: Der Software. Inzwischen hat es sich zu den meisten Entwicklern herumgesprochen, dass eine IP-Adresse nicht nur aus Punkten und Ziffern besteht und maximal 15 Zeichen lang sein darf, dass man sie nicht in einer `uint32` speichern sollte und dass die Darbietung von vier Eingabefeldern für maximal dreistellige Zahlen nicht nur Cut&Paste von IP-Adressen verhindert und damit eine ganz eigene Fehlerquelle darstellt, sondern auch nicht richtig ist. Und auch ssh hat inzwischen gelernt, dass `fe80::98d0:16ff:fe` eine gültige IP-Adresse ist.

Aber lange nicht jede Software geht so souverän mit IPv6-Adressen um. Was schon auf der Client-Seite einer Netzwerkverbindung nicht so ganz rund läuft, wird auf der Serverseite zum ausgewachsenen Drama.

Dass man auf einem Webserver mit Virtual Hosting für jeden virtuellen Server die IP-Adresse hinschreiben muss, ist klar. Aber vielleicht ist nicht ganz so klar, dass damit Autoconfiguration nur noch dann geht, wenn man die IP-Adresse korrekt rät. Als Korollar ist damit auch "Renumbering ist einfach" als Marketinglüge enttarnt. Denn spätestens hier hat einen die Handarbeit wieder eingeholt.

Arbeitet man mit UDP-basierten Protokollen, findet man sich oft in der Situation, alle IP-Adressen des Hosts in der Konfiguration auflisten zu müssen, damit die Software für jede IP-Adresse einzeln einen Socket aufmachen kann, damit ihre UDP-Antworten die richtige Absenderadresse bekommen. Und was beispielsweise unter Linux passiert, wenn man eine naiv programmierte Software starten möchte, die auf einer (noch?) nicht konfigurierten IP-Adresse einen Socket aufzumachen versucht, möge der geneigte Leser selbst ausforschen. Selten ist das Ergebnis das, was man eigentlich haben wollte.

Und nun die Steigerung: Ein Host wechsle zur Laufzeit einer Software die IP-Adresse durch Empfang eines neuen *Router Announcements* und durch *deprecation* der alten IP-Adresse. Was passiert mit einem Listening Socket, der nicht für ::0 aufgemacht wurde?

Software, die IPv6 vernünftig unterstützen möchte, muss darauf vorbereitet sein, dass zu ihrer Laufzeit IP-Adressen erscheinen und verschwinden und dass vielleicht zu dem Zeitpunkt, zu dem sie gestartet wird, noch nicht alle in ihrer Konfiguration erwähnten IP-Adressen benutzbar sind, und sie muss damit einigermaßen souverän und am besten so umgehen, wie der Nutzer es erwartet.

Und es müsste eine Notation für "dieser Hostpart, aber in allen Prefixen, die das lokale System kennt" geben, die man in einer Konfiguration hinschreiben kann, und am besten System Calls, die in der Lage sind, damit umzugehen. Denn sonst muss wirklich jede Änderung an den Netzwerken manuell in der Konfiguration der Software nachgezogen werden, oder man muss einen Mechanismus bauen, um die Konfiguration der Software anhand der aktuell anliegenden IPv6-Netze nachzuvollziehen.

4.6 Privacy aber richtig!

Mit IPv6 wird jeder Host im Internet eindeutig adressierbar. Das ist ein Segen von IPv6, aber auch ein Fluch. Mit einer festen IP-Adresse lässt sich jeder Zugriff eines Hosts eindeutig zurückverfolgen - auch über die Zeit hinweg.

Um dies ein bisschen schwerer zu machen, sind zusammen mit der *Stateless Autoconfiguration* die *Privacy Extensions* definiert. Eine Host mit aktivierten Privacy Extensions wechselt seine IPv6-Adresse in regelmäßigen Abständen. Dabei wird die alte Adresse jeweils *deprecated*, so dass der Wechsel der Adresse bestehende lang laufende Verbindungen nicht beeinträchtigt.

Privacy Extensions nützen etwas, wenn man in einem /64-Netzwerk mit vielen anderen Benutzern ist - zum Beispiel in Firmennetzen oder in größeren Poolräumen in der Universität. In dieser Umgebung können die einzelnen Verbindungen aber auf die Firma, den Poolraum oder gar auf die Reihe zurückgeführt werden, abhängig davon, wie das Netz, in dem der Host angesiedelt ist, strukturiert ist.

Gar nichts nützen Privacy Extensions, wenn man der einzige Benutzer im /64 ist und dieses /64 statisch zugewiesen ist. Dann ist man über den Netzwerkteil der Adresse zurückverfolgbar, unabhängig davon, wie oft die Hosts im LAN ihre IP-Adresse wechseln. Dies wird bei der Internetanbindung für Privathaushalte der Regelfall werden, so dass hier die Internetprovider in der Pflicht sind, die Privatsphäre ihrer Benutzer dadurch sicherzustellen, dass regelmäßig oder wenigstens auf Anforderung durch den Benutzer ein neuer Prefix vergeben wird.

So ein regelmäßig wechselnder Prefix ist natürlich nichts für die Nutzungsfälle, in denen ein Benutzer aus der Ferne auf seine Dienste zugreifen möchte. Das wird entweder wieder eine Anwendung für webbasierte Dyn-DNS-Dienste wie dyndns oder dhis, oder die Provider bieten ihren Kunden die Möglichkeit, sich für einen statischen oder einen dynamischen Prefix oder für beides parallel zu entscheiden. Bisher ist ein so flexibles Endkundenprodukt allerdings noch nicht auf dem Markt aufgetaucht.

4.7 Vorsicht Falle

Manche Dinge, die in einem IPv4-Netz alltäglich sind und funktionieren, sind mit IPv6 nicht mehr so einfach.

4.7.1 Netzwerk durchscannen - nicht wirklich

Zum Beispiel kann man ein IPv6-Netz nicht mehr einfach durchpingen, da ein /64 einfach zu viele Adressen enthält, um sie in endlicher Zeit alle bearbeitet zu haben. Auch ein nmap über das ganze Netz funktioniert nicht mehr. Möchte man wissen, was für Hosts alle im Netz sind, muss man sich die Neighbor Caches (das IPv6-Gegenstück zum IPv4-ARP-Cache) anschauen und hoffen, dass alle Hosts im Netz auch wirklich aktiv sind.

Man ist also gut beraten, sauber zu dokumentieren und den Reverse DNS ordentlich zu pflegen. Sonst kann sich ein Host im Netz verstecken und fällt nicht dadurch auf, dass eine vorher als frei gesehen IP-Adresse plötzlich belegt ist. Die Chance, dass man zufällig genau diese Adresse benutzen möchte, ist besonders bei aus der Autoconfiguration gebildeten IPv6-Adressen doch beliebig gering.

4.8 Autoconfiguration - und jetzt?

Autoconfiguration scheint auf den ersten Blick gut genug dafür zu sein, neue Hosts automatisch ans Netz zu bringen. Leider ist es das auch nur auf den ersten Blick, denn schon die Übermittlung von DNS-Server-Adressen ist eine nicht besonders weit verbreitete Erweiterung. Es gab mal einen vorgeschlagenen Standard, wohldefinierte Adressen für die lokalen DNS-Server zu verwenden, aber diese Idee ist leider nicht aus dem Vorschlagstadium herausgekommen. Deswegen kann man Autoconfiguration mit DHCPv6 ergänzen oder ersetzen und auf diese Weise mehr zentral verwaltete Konfiguration an den Host bringen.

Achtung: Bei Autoconfiguration hat im Gegensatz zu DHCP plötzlich jedes System ein Defaultgateway, also auch das System, das eigentlich nicht mit dem Internet kommunizieren soll. Obacht bei der Firewallkonfiguration ist angesagt.

4.9 Gleich oder nicht gleich?

Softwareautoren finden sich plötzlich in der Situation, dass es sein kann, dass ein Stringvergleich zweier Adressen nicht dieselbe Adresse ist. So sind beispielsweise 2001:db8::1, 2001:DB8::1 und 2001:0DB8::1 dieselbe Adresse. Es ist also notwendig, die Stringdarstellung einer IPv6-Adresse zu normalisieren, bevor man sie vergleicht. Das war zwar auch bei IPv4 bereits notwendig, aber bei IPv4 kam man mit derartiger Schlamperei doch überraschend häufig unbeschadet davon.

4.10 Security-Business as usual

Und Security-Verantwortliche müssen mit IPv6 plötzlich umdenken. Mindestens zwei goldene Regeln, die bisher in jedem Security-Handbuch stehen, müssen für IPv6 umformuliert oder abgeschwächt werden. Die Definition eines mit IPv4 vergleichbaren NAT ist noch so jung, dass sie in vielen Geräten noch nicht umgesetzt ist. Regeln, die besagen, dass interne IP-Adressen nicht im Internet bekannt werden dürfen, zwingen ein IPv6-Netz derzeit dazu, dass mit dem Internet nur noch über Gateways kommuniziert werden kann.

Mit IPv6 wird es zum Regelfall, dass die internen Netze im Internet routebare IP-Adressen haben. Das bedeutet, dass aus dem internen Netz bei fehlkonfigurerter Firewall ein externes Netz werden wird. In einem üblichen IPv4-Setup hat man neben der Firewall noch NAT als zweite Ebene, das verhindert, dass im Fall einer aus Versehen konfigurierten *permit any any* Regel die internen Hosts aus dem Internet erreichbar werden. Bei IPv6 hat man diese zweite Ebene nicht, hier schlägt der erste Fehler voll durch.

4.11 ICMP

ICMPv6 ist im Gegensatz zu ICMPv4 integraler Bestandteil der Kommunikation. Unter anderem ist die Zuordnung zwischen IP-Adresse und MAC-Adresse des Ziels bei IPv6 im Rahmen von ICMPv6 definiert, während ARP für IPv4 ein ganz eigenes Protokoll ist. Wenn man also - wie bei IPv4 leider üblich - ICMP komplett sperrt, merkt man das relativ schnell, weil IPv6 ohne ICMPv6 überhaupt nicht funktioniert.

Dadurch, dass ein Host über Autoconfiguration in aller Regel nur eine Defaultroute lernen kann, werden ICMP-Redirects bei IPv6 wichtiger als bei IPv4, so dass man gerne haben möchte, dass ein Host ein Redirect, das von einem Router im LAN stammt, beachtet und ausführt. Nur so kann in komplexeren Netzwerken der kürzeste Netzwerkweg gefunden werden.

4.12 Ungewolltes IPv6 per Tunnelmechanismus

Manche Betriebssysteme (z.B. aktuelle Windows-Varianten) versuchen unter Umständen, sich mit aller Gewalt IPv6-Konnektivität zu beschaffen. Hierzu benutzen sie Tunneltechniken wie Teredo, was unangenehme Folgen für die Security haben kann. Abhilfe schafft hier, zu wissen was man einsetzt und diese Mechanismen im Betriebssystem zu deaktivieren. Sites, die auch für ausgehenden Traffic eine restriktive Firewall-Politik fahren und nur das erlauben, was explizit gewünscht ist, sind hier auf der sicheren Seite.

5 Kasten: Dynamischer DNS

Dynamischer DNS macht schon das Leben mit IPv4 sehr viel einfacher. Bei IPv6 wird es dank Autoconfiguration noch viel wichtiger, weil sich die IP-Adressen eines Hosts häufiger ändern und man viel des IPv6-Komforts verliert, wenn man manuell den DNS nachziehen muss.

In diesem Punkt ist die Windows-Welt tatsächlich mal im Vorteil gegenüber uns Unixern: Größere Windows-Installationen haben üblicherweise ein Active Directory, und eins der hervorstechenden Merkmale des Active Directory (*AD*) ist, dass sich alle im AD integrierten Hosts automatisch beim DNS anmelden und dort passende und richtige DNS-Einträge bekommen. Das funktioniert bei richtiger Konfiguration vorwärts und rückwärts, für IPv4 und IPv6 - weltweit aber nur, wenn man sich traut, die Zonen auch wirklich auf die Windows-Domaincontroller zu delegieren. Für den Forward-DNS kommt hinzu, dass man die Domain, in der man das AD betreibt auch "in der Welt" besitzen muss.

Aber es ist immerhin mehr, als wir Unixer üblicherweise haben. Bei uns wird Dynamic DNS für IPv4 von bind und dem ISC-DHCP-Server unterstützt. Hier ist es normalerweise die Aufgabe des DHCP-Servers, von ihm vergebene IP-Adressen beim DNS einzutragen. Das erfordert natürlich, dass der Host mit DHCP betrieben wird - was er ja auch meistens ist, denn sonst hätte er ja eine statische IP-Adresse und einen statischen DNS-Eintrag. Das Tooling, um vom Host aus selbst DNS-Einträge zu erzeugen, ist mindestens in den gängigen Linux-Distributionen vorhanden; mit der Integration in die Mechanik der Netzwerk-Initialisierung hapert es jedoch.

Mit dem aufkommen von Dynamic DNS muss aber auch das Vertrauen in DNS-Einträge im Security-Kontext abnehmen: Wenn ein kompromittierter Host DNS-Einträge vornehmen kann, bietet dies interessante neue Angriffsvektoren.

Reboot reloaded – Patching the Linux kernel without reboot

Dr. Udo Seidel
Amadeus Data Processing GmbH
Berghamer Strasse 6
D-85435, Erding
BR Deutschland
<udoseidel@gmx.de>

1 Abstract

Changes of the Linux kernel are quite normal. In case of security updates there is even a tight deadline for the implementation. The standard process for binary kernel patches is the installation and activation of a new one. That activation requires a reboot of the entire OS. This might not be desired for various reasons. Ksplice promises to be a solution for that catch-22.

2 The theory

2.1 Some history

Like many other projects of the opensource and Linux community Ksplice was started by students. Here, back in 2008, we have four students at the MIT and the diploma thesis of Jeff Arnolds who was the later CEO of Ksplice Inc. While software was released under the GPLv2 the company was founded to provide services around that technology. Back the quite a few Linux distributions were supported, e.g. Red Hat, Fedora, CentOS, Debian and Ubuntu. In 2011 Ksplice Inc. was acquired by Oracle and things have changed since then. For some time only paying customers received the online kernel patches. Existing customer were served as well via a 'legacy' model but the ongoing development of the Linux kernel made this business case obsolete. In the recent past the Ksplice services were opened up a little bit and now Fedora and Ubuntu users can happily use it.

2.2 First thoughts

Ksplice focuses on security fixes of the Linux kernel. The source code of such fixes are often only a few lines long. Secondly, closing security holes has often a tight deadline and hence complicates the scheduling of a reboot. Therefore, security fixes are good candidates for online patches.

There are several approaches to create online patches. An obvious one is based on the comparison of the source code. This looks simple in the first place but requires the corresponding programming language skills. The analysis of the source is not straight forward either. Changes in the source in one region of the code can have knock-on effects in other parts as well. Again, high programming skills are crucial to master that challenge. Lastly, knowing what to change in the source code does not answer the question how to get this into the running kernel of the systems in scope.

Ksplice uses a different approach and does a comparison on object code level. Here, there is much less need for programming language skills. Also, the code analysis is implicit per design. A positive side-effect is that automation of comparison and code analysis seems to be possible. However, the

question how get the new code active in the kernel is not answered. Also, more details are needed on the generation of the object code.

2.3 Preparation is key

Online patching consists of two steps: the generation of the patches and the actual patching itself. For the first step the original kernel sources, the changed source code and an kernel build environment is needed. Ksplice generates object out of the original kernel sources and out of the changed one. Lets call them PRE and POST object code. Ksplice compares PRE and POST and generates a delta. This result is the one which contains the changed code to be used by the kernel. Comparing object code is easier said than done – has its challenges and opportunities. The object code analysis is done via the BFD library which is used for core file investigations too. Secondly, Ksplice limits address information in the code and tackles that at a later stage. Thirdly, it generates object code where each data structure and function has its own section. The GNU C compiler provides the options `-ffunctions-sections` and `-fdata-sections` for that purpose. Per design the delta code will contain changes which are not or only indirectly caused by the change of the source code. This is almost a given if the used compiler differs from the one which used to generate the running kernel in the first place. It can happen too if there non-local optimisations done by the compiler.

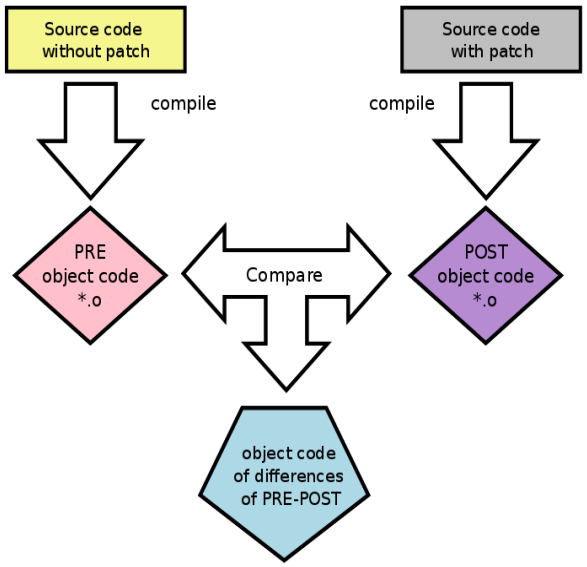


Figure 1: Generation of the delta code by comparing PRE and POST object code.

Ksplice has no chance to detect unnecessary code replacement. However, since the replacing code is not changing anything there are not functional drawbacks. Just an increased footprint regarding memory and CPU cycles which can be usually neglected.

2.4 Where to deliver the new stuff

As mentioned before, the delta code does not contain any address information. Per design it refers to symbols. Is not clear either, which memory addresses point to code which should be replaced. The task to resolve symbols to memory addresses is not unique and quite standard while performing core dump analysis. In the case of the kernel there is even a table which could very handy here. Unfortunately, there are symbol names which are not unique. In order to solve this challenge Ksplice

re-uses the PRE code. This was generated without address information too. However, there is a counterpart of it already loaded into the memory by the current kernel – let's call this RUN code. By performing a PRE-RUN comparison Ksplice is able to filter out the 'wrong' symbols and extract the needed memory information. Again, this is easier said than done. The tools needs some architecture-specific information like instruction length for that.

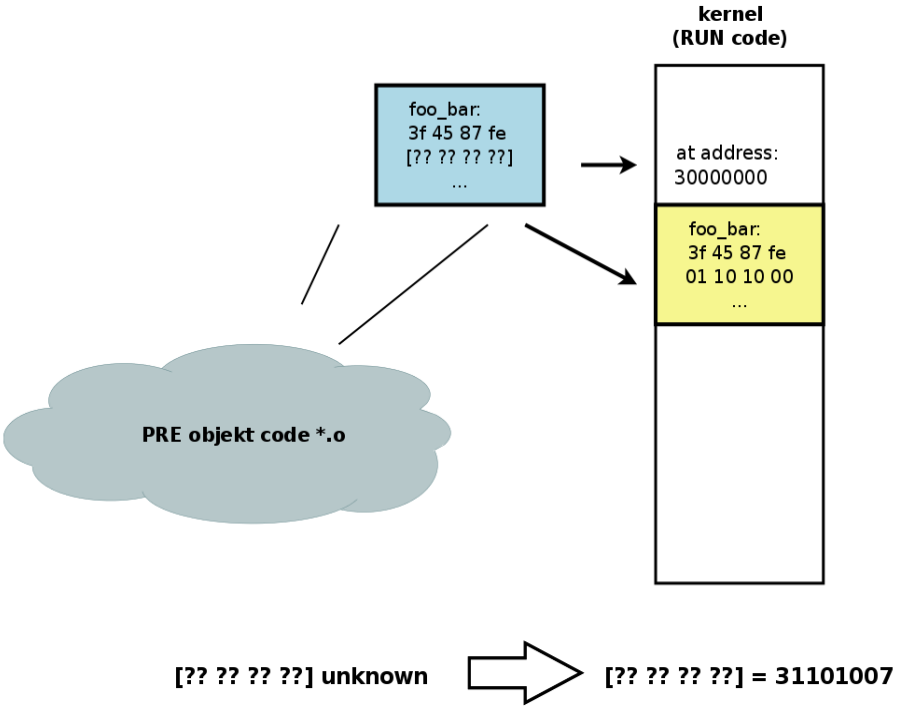


Figure 2: Determination of the memory address comparing PRE and RUN object code.

Now it is known, where the code is located which should be replaced. Ksplice is very cautious during the address determination. If something unexpected happens the PRE-RUN comparison is canceled.

2.5 Putting things together

The final step is put all information together and make it usable by the kernel. Ksplice does some post processing of the delta code which results in a kernel module, e.g. *ksplice-new.ko*. Secondly, the kernel has to know to use the new code instead of the current one. This is achieved by a second kernel module *ksplice.ko* which contains jump functions. This will deviate from the original memory address, which is known from the PRE-RUN comparison, to the one of the newly loaded kernel module. In a nutshell: each source code patch will materialize in two kernel modules to be generated and loaded by Ksplice.

2.6 Final thoughts

The Ksplice approach does not come limit-free. The bigger the patch the more complex the several comparisons are. Also, if the patch includes semantic changes, additional manual coding is needed to add the new items into the running kernel. Lastly, accumulating Ksplice generated patches is possible

and also desired to a certain extent. However, the assumption is that sooner or later the online patched kernel is replaced by 'normal patched' one via a reboot. From an operational point of view there are even more questions. How industrialize the online patching? How to compare a Ksplice patched kernel with the other kernel?

3 The operations point of view

3.1 First remarks

In order to be maintainable and operable Ksplice uses a client server approach. The server part is responsible for providing the patches and makes them available via software repository. A desired side-effect of that approach is the possibility to create dependencies. This is used to make the patched kernel comparable to normal kernel by applying additional patches. The client side is responsible for downloading and applying the patches. The client side decides if the patches are reboot-resistant or not. Since some kernel modules are part of the initial ramdisk OR the code has to be loaded as early as possible, the Ksplice client hooks into the normal tools to load and unload kernel modules.

Command	Description
uptrack-install	Install available patch(es)
uptrack-remove	Remove installed patch(es)
uptrack-show	Status of installed and/or available patches
uptrack-uname	Effective/patched kernel version
uptrack-upgrade	Install ALL available patches

Table 1: Commands of the Ksplice client which runs on the server which will be patched online

3.2 The tool

At the moment there is only one provider of Ksplice kernel patches and this is Oracle. The corresponding client contains the software repository information for accessing and downloading the patches. If you have an Ubuntu or an Fedora Linux you can access the patches without any costs. The same is true for Oracle Linux customers or RHEL admins who have paid for that support. As outlined before, the client will replace some commands dealing with kernel modules. The original commands will be preserved and chain-executed.

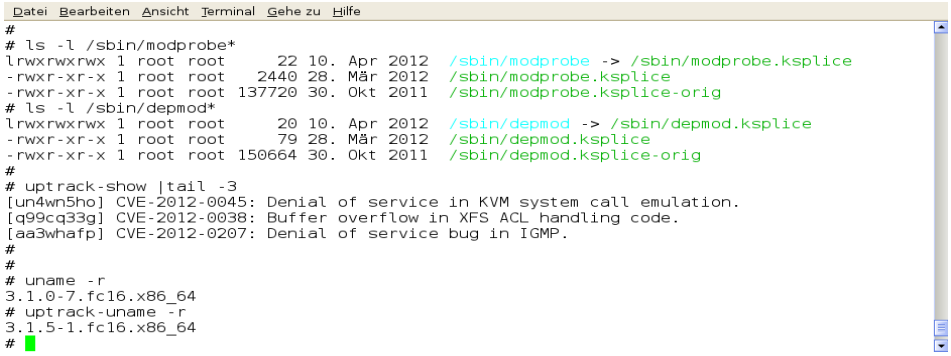


Figure 3: Ksplice client manipulates some kernel related commands.

3.3 Real world example

A very good example is the IGMP bug which was introduced in kernel 2.6.36. The CVE reference number is 2012-0207. The risk of impacted kernels is very high because the bug can be exploited via the network and results in a kernel panic. The fix itself is a two liner:

```
diff --git a/net/ipv4/igmp.c b/net/ipv4/igmp.c
--- a/net/ipv4/igmp.c
+++ b/net/ipv4/igmp.c
@@ -875,6 +875,8 @@ static void igmp_heard_query(struct in_device *in_dev, struc
t sk_buff *skb,
                  * to be intended in a v3 query.
                  */
                 max_delay = IGMPV3_MRC(ih3->code)*(HZ/IGMP_TIMER_SCALE);
+                if (!max_delay)
+                        max_delay = 1;  /* can't mod w/ 0 */
         } else { /* v3 */
                 if (!pskb_may_pull(skb, sizeof(struct igmpv3_query)))
                         return;
```

All in all an ideal candidate for Ksplice.

4 Conclusion

The Ksplice approach is very promising. It release the pressure to reboot systems within 30 days in order to apply security patches. The tool is cautious and prefers to rather not apply a change if it is too risky. The current operational model, e.g. making the patched kernel comparable to a distribution kernel, works as well. A open question is what Oracle plans for the tool, especially since it is released under the GPL

References

[1] http://www.ksplice.com/

[2] http://www.ksplice.com/doc/ksplice.pdf

[3] http://kerneltrap.org/mailarchive/linux-kernel/2008/4/23/1570474

[4] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0207

Netzmanagement mit HIRN

Robin Schröder
Rechenzentrum der Ruhr-Universität Bochum
Universitätsstr. 150
44801 Bochum
Deutschland
<robin.schroeder@ruhr-uni-bochum.de>

1 Abstract

Die Ruhr-Universität Bochum (RUB) ist seit mehr als 10 Jahren vollvernetzt. Das HIRN („**H**ochschul-**I**nternes **R**echner-**N**etz“) erstreckt sich über den Campus sowie viele von der Universität in der Stadt angemietete externe Standorte und wächst ständig. Jeder der mittlerweile mehr als 60.000 Netzwerkanschlussports in Benutzerzugriff befindet sich in einem von über 900 VLANs und ist durch das Network Operation Center managebar. Dabei wird Hardware von zur Zeit drei verschiedenen Herstellern eingesetzt. Das Management erfolgt mit selbst entwickelter Software, die über Jahre gewachsen ist und auch Benutzern zahlreiche Möglichkeiten bietet, per Web-Interface auf ihre Netzkonfiguration Einfluss zu nehmen.

In diesem Artikel wird zum einen aus der Praxis berichtet: Wie ist das Datennetz der RUB aufgebaut? Wie werden Geräte verschiedener Hersteller mit einer einzigen Software „Hersteller-Transparent“ gemanaged? Wie wird die Dokumentation aktuell gehalten? Was passiert, wenn man „von Hand“ an den Geräten konfiguriert? Was sind typische Fallstricke?

Zum anderen soll der geneigte Leser hier Anregungen finden zur Netzstrukturierung im Allgemeinen, beispielsweise den Umgang mit dem Problem *Masse* (hier in Bezug auf die Anzahl der Netzkomponenten und Verarbeitung von statistischen Daten), die Vor- und Nachteile des Verzichts auf herstellergebundene Managementsoftware sowie vieles weitere.

2 Strukturelles

2.1 Aussenanbindung

Das Datennetz (HIRN) der Ruhr-Universität Bochum verfügt neben einer 10-Gigabit-Anbindung zum DFN-Verein auch noch über eine 2-Gigabit-Anbindung zum in Bochum ansässigen Provider TMR. Weiterhin gibt es redundante 10-Gigabit-Verbindungen zu den Universitäten Dortmund und Duisburg-Essen. Die drei Universitäten betreiben im Rahmen der „Universitätsallianzmetropole Ruhr“ (UAMR) IT-Dienste verschiedener Schwerpunkte für jeweils alle drei Universitäten.

2.2 Backbone

Die RUB verfolgt das Konzept des dezentralen Routings, d.h. in (fast) jedem Gebäude gibt es im Gebäudehauptverteiler, in dem sich neben der Zusammenführung der Netzverbindungen des Gebäudes oft auch TK-Infrastruktur befindet, einen Router. Dieser ist per 10GBit/s Monomode Glasfaser mit dem zentralen Backbone im Rechenzentrum verbunden.

2.3 Netzbetreuer

Es werden bei Lehrstuhlnetzen (IP-Subnetze, Broadcast-Domains) lokale „Netzbetreuer“ als Ansprechpartner benannt. Diese sind zuständig für die lokale Zuordnung von IP-Adressen aus einem vom Rechenzentrum überlassenen Adresspool zu den einzelnen Geräten der Einrichtung, müssen jedoch nicht über tieferes Netz-Know-How verfügen. Gleichzeitig sind die Netzbetreuer auch der erste Ansprechpartner seitens des Network Operation Center (NOC) im Falle von Störungen und Problembehebungen. Den lokalen Ansprechpartnern werden zum Management ihrer IP-Subnetze durch das NOC sowie die zentrale Netzverwaltung im Rechenzentrum administrative und operative Hilfsmittel in Form von Web-Interfaces zur Verfügung gestellt, die eingeschränkt auf ihren Verantwortungsbereich verwendet werden können (DNS, DHCP Service, Analysetools, MAC-/IP-Adressfinder, div. Schaltungsmöglichkeiten auf Port-Ebene).

2.4 Network Operation Center

Das HIRN wird durch das dem Rechenzentrum angegliederte Network Operation Center[1] (NOC) betrieben, welches zur Zeit aus einer Abteilung des Rechenzentrums mit sieben Personen besteht. Es werden im Kern die folgenden Bereiche abgedeckt:

Unter dem Bereich *Netz physisch* verstehen sich Hardwarearbeiten wie die Installation von Switches und Access Points, die Konfiguration von Edge-Ports sowie die Fehlerbehebung vor Ort. Dazu gehört auch die entsprechende Grundkonfiguration neuer Geräte sowie die Restauration von Konfigurationen aus dem Backup im Falle eines Hardwaretauschs.

Abbildung 1: Ein Blick auf unser *cold standby*: Teilweise gebrauchte Geräte vieler Marken und Serien

Dabei setzen wir auf das Prinzip „keep it simple“, d.h. wir haben in der Regel keine redundanten Uplinks und kein failover oder *hot standby*. Unser „*cold standby*“ sind ein paar Euro-Paletten voll mit Switches aller Art (Siehe Abb. 1), die wir mal irgendwo verbaut haben. Auch Core-Switches/Router haben wir in Reserve. Im Fehlerfall tauschen wir dann die Hardware möglichst schnell, d.h. der betroffene

[1] http://noc.rub.de/

Teilbereich des Netzes funktioniert in der Regel innerhalb von maximal einem Tag[2] wieder. Ausnahmen können hier seit dem Bestreben der Universität, auch Flächen ausserhalb des Campus anzumieten, die angemieteten LWL-Strecken zu den externen Standorten bilden. Beispielsweise bei Baggerarbeiten. Aber selbst in einem solchen Fall konnten wir bereits einmal mit einer WLAN-Verbindung aushelfen.

Durch den flächendeckenden, konsequenten Einsatz voll managebarer Hardware muss im Falle von Umzugsarbeiten oder Netzumschaltungen normalerweise keiner „zum Patchen“ an die betroffenen Standorte laufen. Die Port-Konfiguration kann in manchen Fällen sogar direkt vom Netzbetreuer selbst per Web-Interface erfolgen, ganz ohne die Beteiligung des NOC. Anderenfalls klären wir in der Regel telefonisch oder per E-Mail erforderliche Umschaltungen im Vorfeld mit den Netzbetreuern ab.

Der Bereich *Netz Logisch* umfasst den größten Teil der Softwareentwicklung für den Eigenbedarf sowie die Konfiguration der Core Switches und Router, insbesondere im Hinblick auf dynamische Routingprotokolle (wir haben OSPF zwischen unseren Gebäudeverteilern und BGP für die Aussenanbindungen). Dazu gehört auch die Vergabe von VLANs (Broadcast-Domains) und IP-Subnetzen an Insitutionen der RUB, welche dem NOC jeweils einen oder mehrere Netzbetreuer als Ansprechpartner zur Verfügung stellen.

Im Bereich der *Dokumentation* wird eine wichtige Grundlage für die Funktionalität des Datennetzes geschaffen. An dieser Stelle wird dafür gesorgt, dass Fotos aller Standorte sowie Grundrisspläne und Messprotokolle vorhanden und an den richtigen Stellen in unserer Software verlinkt sind, dass Netzbetreuerdaten gepflegt werden und die Rechteverteilung aktuell gehalten wird. Insbesondere die Pflege der Netzbetreuerdaten ist an einer Universität deshalb nicht zu unterschätzen, weil es keinen direkten Informationsfluss zwischen beispielsweise Personalabteilung und IT-Management gibt. Wir bekommen also im Normalfall nicht mit, wenn sich etwas verändert, z.B. ein Professor wechselt, dadurch ein Lehrstuhl umbenannt wird, sich infolge dessen dort unsere Ansprechpartner ändern oder ähnliche Dinge geschehen. An dieser Stelle ist man auch zum Teil etwas auf den „Flurfunk“ und die Mitarbeit aller Beteiligten angewiesen.

Zu guter letzt betreiben wir auch noch das *Mailsystem* der RUB, was jedoch nicht Thema dieses Artikels ist.

2.5 H.I.R.N.-Ports

Im Datennetz der RUB finden sich etwa 5.500 gekennzeichnete (Siehe Abb. 2) sogenannte *H.I.R.N.-Ports*. Das sind Datennetzanschlüsse, an denen der Nutzer per DHCP eine Netzkonfiguration zugewiesen bekommt und dann nach Freischaltung seiner IP-Adresse per Authentifizierung auf einer Login-Seite einen weitgehend uneingeschränkten Netzzugang bekommt.

Abbildung 2: Fast überall sind Netzanschlüsse zu finden, die mit diesem Symbol gekennzeichnet sind.

[2] wobei hier gilt: Tag = Arbeitstag (5x8, nicht 7x24)!

2.6 eduroam

Die RUB nimmt mit dem auf dem Campus an zahlreichen Stellen verfügbaren WLAN am eduroam[3] Projekt teil. Zur Zeit sind knapp 300 Access Points im Einsatz, Tendenz steigend. Leider lassen die finanziellen Mittel eine flächendeckende WLAN-Versorgung nicht zu, weshalb nur Bereiche in und um Hörsälen, Seminarräumen sowie Cafeterien und andere stark frequentierte Bereiche versorgt werden können.

3 Eigenentwicklungen

Warum wir den Kern unserer Netzmanagementsoftware selbst programmiert haben? Das hat mehrere Gründe. Viele der angebotenen Softwareprodukte haben das „Problem der Masse“, d.h. sie kommen entweder schon allein mit der Anzahl unserer Geräte nicht zurecht, oder aber sie schwächeln beispielsweise bei der Datenverarbeitung, weil sie zu ressourcenhungrig sind. Es gab auch vor mehr als 10 Jahren nichts, was alle unsere Ansprüche hinreichend erfüllt hat. Desweiteren sind wir mit eigener Software in der Lage, extrem schnell gezielt zu erweitern und zu bugfixen.

3.1 Port Descriptions

Der Schlüssel zu unseren Management-Tools sind die Port Descriptions (*Port-Text*), die jedes von uns eingesetzte Gerät beherrscht. Jeder Switchport im Datennetz der RUB bekommt mit dem Parameter `description` (bzw. `name` auf HP-Geräten) eine Beschreibung. Diese Beschreibung ist nicht unbedingt ohne Weiteres gut lesbar, enthält jedoch codiert alle Informationen, die wir für eine Zuordnung zu Gebäude, Etage, Raum, Anschlussbeschriftung, Patchfeld und Patchfeld-Port benötigen. Dabei ist zu bedenken, dass auf vielen Geräten die Port-Description maximal 64 Zeichen umfassen darf.

In der Historie wurden an dieser Stelle viele verschiedene Port-Description-Formate verwendet, die teilweise noch immer im Einsatz sind, so dass der Port-Text-Parser im Backend des *HirnMS* mittlerweile recht komplex ist. Das momentan gebräuchliche Port-Text-Format sieht beispielsweise folgendermaßen aus:

```
A;Temprack4;D;14;03-252;5;NAF/03/252
```

Darin verbirgt sich, jeweils durch Semikolon getrennt, folgende Information:

- **A:** Port-Text-Identifier (hier ‚A‘, weil dies das erste Format ist, welches einen Identifier hat)
- **Temprack4:** Verteiler-Standort
- **D:** Patchfeld (i.d.R. von A-Z durchnummeriert)
- **14:** Patchfeld-Port (1-16, 24 oder 32, je nach Patchfeld-Typ)
- **03-252:** Anschlussbeschriftung im Raum (Beschriftung der Dose im Büro)
- **5:** Fortlaufende Port-Nummer im Raum
- **NAF/03/252:** Gebäude/Etage/Raum des Anschlusses (anklickbar im Web-Frontend)

Gegebenenfalls können, durch ein weiteres Semikolon getrennt, noch Freitext-Informationen folgen, beispielsweise die Information, dass an diesem Port nur 100 MBit/s funktionieren wegen schlechten Kabelmaterials o.ä.

Ältere Port-Text-Formate enthielten beispielsweise nur `Gebäude/Etage/Raum/Port` oder aber auch noch zusätzliche Informationen: `Gebäude/Etage/Raum;Verteiler/Patchfeld/Port`.

[3] http://www.eduroam.org/

Der Prozess der Port-Text-Entwicklung schreitet bei uns jedoch kontinuierlich voran, so dass dieser durchaus weitere Änderungen erfahren wird.

Aus den Port-Texten gewinnen wir also die Informationen, wo sich ein Anschluss befindet. Wir pflegen natürlich auch eine Datenbank, in der wir diese Informationen nachhalten. Bei Änderungen am Port-Text durch unser Frontend *HirnMS* werden diese automatisch synchronisiert. Nach manuellen Änderungen direkt am Switch liest der Aufruf des Programms `syncswitchtodb.pl` die komplette Port-Konfiguration eines Switches in die Datenbank ein, wobei nur die Änderungen übernommen werden, falls das Gerät schon einmal eingelesen wurde. Dabei wird gegebenenfalls die Port-Raum-Zuordnung anhand des Port-Textes oder der SNMP-Location des Geräts aktualisiert.

3.2 Frontend: HirnMS

Für die bequeme Konfiguration von Standardaufgaben (Konfiguration von Edge-Ports, Bridgen von Vlans, Ändern von Port-Descriptions etc.) haben wir vor einigen Jahren eine Webbasierte Software in PHP mit einer MySQL-Datenbank geschrieben, das *H.I.R.N. Management System* (HirnMS). Diese Software existiert bis heute, hat vom Look & Feel her aber den „Charme der Jahrtausendwende“ behalten.

Wir arbeiten bis heute mit dem HirnMS als zentrale Software zur Konfiguration unseres Netzes. Mittlerweile wurde das alte, in PHP geschriebene HirnMS-Backend jedoch durch ein neues Backend ersetzt.

3.3 Backend: HIRN-Module

Zur Kommunikation mit unseren aktiven Geräten (Router und Switches) haben wir als Abstraktionsebene – manche mögen auch *Middleware* dazu sagen – eine Schicht Perl-Module geschrieben. Mit diesem objektorientierten Ansatz gelingt es uns, über verschiedene Wege direkt mit den aktiven Komponenten zu kommunizieren, je nach Hersteller und Modell, wobei der eigentliche Funktionsaufruf immer neutral für alle Geräte ausprogrammiert ist. Vererbung sorgt dabei in den Perl-Modulen für recht übersichtlichen Code.

So liefert z.B. das Attribut `neighbors` auf einem Gerät (genauer: auf dem Objekt, welches ein Gerät repräsentiert) immer die per CDP und/oder LLDP ermittelten benachbarten Geräte in Form einer Referenz von Hashes.

3.3.1 Verbindungsaufbau

Folgendes Perl-Script baut eine Verbindung zu einem Gerät (hier `pc5500-test-1`) auf und gibt die Neighbors zurück:

```
#!/opt/perl/bin/perl

use HIRN;
use Data::Dumper;

my $dev = HIRN->hirndevice_for_lifedev('pc5500-test-1') or die "no connection";
print Dumper $dev->neighbors();
```

Die Ausgabe (mit `Data::Dumper` dargestellt) sieht dann beispielsweise so aus:

```
$VAR1 = {
          '10.11.1.2' => {
                           'local_port' => 'GigabitEthernet1/0/47',
                           'remote_port' => 'GigabitEthernet1/0/48',
                           'device' => 'pc5500-test-2'
                         },
          '10.11.0.1' => {
                           'local_port' => 'GigabitEthernet1/0/48',
                           'remote_port' => 'GigabitEthernet12/4',
                           'device' => 'cat6500-rz-1'
```

```
                        }
    };
```

Dabei ist es unerheblich, um was für eine Art Gerät es sich handelt. Beim Erzeugen des Device-Objekts (`$dev` in obigem Beispiel) wird zunächst per SNMP die `sysObjectID` (`1.3.6.1.2.1.1.2.0`) ausgelesen und anhand des Werts eine entsprechende Klasse ausgewählt. Das sieht bei uns in etwa so aus, wobei `$oid` hier die ausgelesene `sysObjectID` ist:

```
$oid =~ s/^\.?1\.3\.6\.1\.4\.1\./enterprises./;
$oid =~ s/\./_/g;
$classname = 'HIRN::Device::' . $oid;
$classname = 'HIRN::Device::Cisco' if $oid =~ /^enterprises_9_/;
$classname = 'HIRN::Device::HP'    if $oid =~ /^enterprises_11_/;
$classname = 'HIRN::Device::Dell'  if $oid =~ /^enterprises_674_/;
```

Anschließend wird ein Objekt vom Typ `$classname` initialisiert und zurückgegeben.

Dieses Verfahren ermöglicht es uns, relativ generisch gehaltene Klassen für z.B. Cisco IOS vorzuhalten (`HIRN::Device::Cisco`), auf Eigenarten spezieller Geräte oder Software-Versionen jedoch in „tieferen“ Klassen einzugehen, indem dort Funktionen von `HIRN::Device::Cisco` überschrieben werden.

So gibt es zum Beispiel die Klasse `HIRN::Device::enterprises_9_1_618` für Cisco Wireless Access Points, in der unter anderem die Funktion `uplink_port()` überschrieben wird. Sie gibt für diese Access Points immer `FastEthernet0` zurück. Warum? Weil es nur eine Ethernet-Schnittstelle gibt und unser Verfahren, auf Cisco-Geräten den Uplink-Port – also den Port, der in Richtung Router zeigt – zu finden, auf den Access Points nicht funktioniert.

3.3.2 Telnet statt SSH

Beim ersten Aufruf einer Funktion nach dem Erzeugen des Device-Objekts wird per Telnet eine Verbindung zum Device hergestellt und die Login-Prozedur abgearbeitet, so dass alle weiteren Funktionen ausgeführt werden können. Telnet? Ja. Hier treffen wir – wie an zahlreichen Stellen des HirnMS – auf das Problem der Masse. Telnet ist viel schneller als SSH und der zeitliche Mehraufwand von 2-3 Sekunden für den SSH-Login macht sich bei rund 2.000 Geräten schnell bemerkbar. Denn die HIRN-Module werden natürlich auch von Automaten benutzt, die sich regelmäßig aus verschiedensten Gründen auf den Switches einloggen und Informationen abfragen. Unser Management ist ausreichend geschützt und gut überwacht, so dass wir nicht befürchten müssen, dass die Nutzung von Telnet statt SSH unsere Infrastruktur nennenswert in Gefahr bringt.

3.3.3 Einige Funktionen

Nach dem Erzeugen eines Device-Objekts können unter anderem folgende Attribute von jedem Device abgefragt werden:

- `interface_information` Gibt Informationen (Port-Status, Speed) über alle oder ein bestimmtes Interface zurück
- `interfaces` Gibt eine Liste der Interfaces zurück
- `vlans` Gibt eine Liste der VLANs (VLAN-ID und Name) zurück
- `location` Gibt die auf dem Gerät eingestellte SNMP-Location zurück
- `uplink_port` Gibt den Uplink-Port zurück
- `neighbors` Gibt die per CDP und/oder LLDP erkannten Geräte zurück

Interessant in diesem Zusammenhang ist die SNMP-Location. Sie hat bei uns – ähnlich wie die Port Description (siehe weiter unten) – ein festes Schema. Im Zweifelsfall, d.h. wenn die Port-Description es nicht hergibt, wird die SNMP-Location benutzt, um eine Gebäude/Etage/Raum-Zuordnung eines Switchports zu realisieren.

Weiterhin gibt es jede Menge Aktionen, die mit einem Device ausgeführt werden können. So z.B.:

- `access_port()` Schaltet einen Access-Port mit einem ungetaggten VLAN
- `dot1q_port()` Erzeugt einen 802.1q Trunk Port mit mehreren getaggten VLANs
- `ap_port()` Erzeugt einen 802.1q Trunk Port, so wie wir ihn für Access Points benötigen
- `vlan_add()` Fügt ein VLAN zu einem 802.1q Trunk hinzu
- `vlan_remove()` Entfernt ein VLAN von einem 802.1q Trunk Port
- `port_text()` Setzt bzw. ändert eine Port-Description auf einem Interface
- `save_configuration()` Speichert die Konfiguration auf dem Device
- `create_vlan()` Erzeugt ein VLAN auf dem Device
- `add_vlan_to_switch()` Legt ein VLAN auf dem Device an und verbindet es durch Bridging über die Uplink-Ports vom Router durch über alle dazwischen liegenden Geräte bis zum Device

Hier sind insbesondere die Aktionen `port_text()` und – beispielhaft für eine ganze Reihe an Aktionen, die nicht nur das gerade gewählte Device betreffen – `add_vlan_to_switch()` zu nennen. Die Funktion `port_text()` bekommt ein Interface sowie einen String als Parameter. Letzterer ist entweder die neue Port Description oder ein regulärer Ausdruck. Wird der String als regulärer Ausdruck erkannt, wird er auf die bereits bestehende Port Description angewendet, anstatt sie zu ersetzen. Das ermöglicht uns Bulk-Änderungen an den Port Descriptions.

Die Funktion `add_vlan_to_switch()` findet live im System heraus, wo die *root bridge* des hinzuzufügenden VLANs ist und fügt das VLAN dann auf allen Trunk-Ports von der root bridge aus bis zum ausgewählten Device hinzu. Dazu werden unter anderem die Attribute `neighbors` und `uplink_port` benutzt. Der Aufrufer der Funktion muss keine nähreren Informationen dazu kennen.

Es gibt viele weitere Aktionen, die jedoch den Umfang dieses Artikels sprengen würden.

3.4 Port-Statistiken

Wir sammeln von jedem Switchport alle fünf Minuten die Interface Counter per SNMP ein und merken sie uns mit RRDtool in Round-Robin-Datenbanken. Bei knapp 65.000 Ports und bis zu sieben Werten pro Port (`inoct`, `outoct`, `inerr`, `outerr`, `indisc`, `outdisc`, `inbcast`) sind das gut 450.000 Werte. In fünf Minuten so viele Werte per SNMP lesen ist möglich, nicht aber, sie in 65.000 Dateien zu schreiben. Das Anfassen so vieler Dateien inklusive des RRDtool processings ist zu viel für gängige Hardware-Konstruktionen.

Wir nutzen zur Datensammlung die jeweils aktuelle Version von NeDi[4] in einer leicht modifizierten Variante mit einer 40 Gigabyte großen RAM-Disk (knapp 20 Gigabyte sind zur Zeit belegt), auf der die RRD-Files liegen. Mit der RAM-Disk haben wir das Problem des Dateizugriffs gelöst. Alle paar Minuten machen wir mit `tar` ein Backup auf Festplatte, allerdings ohne Kompression. Beim Booten der Maschine wird das letzte Backup auf die RAM-Disk entpackt. Damit entstehen zwar Lücken in den Datenreihen, die statistisch gesehen jedoch vernachlässigbar sind.

[4] http://www.nedi.ch/

Viel problematischer ist, dass NeDi für einen Durchlauf über unsere rund 2.000 Geräte momentan bis zu 10 Stunden benötigt. Dabei wird allerdings auch jede Menge mehr erfasst, als nur die interface counter. NeDi hat Seriennummern aller Geräte und Transceiver, CDP oder LLDP neighbors, Software- und Firmware-Versionen und vieles weitere. Diese Daten ändern sich nicht allzu oft, so dass es uns reicht, sie alle 12 Stunden zu erfassen. Bei einem vollständigen NeDi-Durchlauf werden dann auch neue Geräte gefunden. Diese Informationen wiederum machen wir uns an vielen anderen Stellen zunutze, denn NeDi legt sie in einer MySQL-Datenbank ab.

Wir haben daher eine Erweiterung programmiert, die lediglich alle fünf Minuten parallelisiert per SNMP die interface counter abfragt und sie in die RRD-Files von NeDi schreibt. Dazu benötigt unser Tool ca. drei Minuten. Andere zeitintensive Dinge lassen wir aussen vor.

Ein weiteres Problem gestaltete sich dahingehend, dass NeDi nur CDP oder LLDP, nicht aber beides benutzt – sofern das Gerät dies unterstützt – um die neighbors zu erkennen. Wir haben viele Geräte, die beides können. HP- und Dell-Hardware kann jedoch – zumindest teilweise – kein CDP, ältere Cisco-Hardware kann kein LLDP. Und noch schlimmer: Die Core Switches der Cisco Catalyst 6500 Serie können beides, liefern per SNMP jedoch keine LLDP-Daten. LLDP-Informationen können wir dort nur per Telnet abfragen. Ein kleiner Patch sorgt dafür, dass NeDi CDP und LLDP benutzt, wenn ein Gerät es beherrscht.

Das Webinterface von NeDi hat eines dieser typischen „Masse“-Probleme: Die Auswahlliste der Geräte ist eine DropDown Box. Mit 20 oder vielleicht 50 Geräten mag das noch schön sein, nicht aber mit 2.000. Zur Auswertung der per NeDi gesammelten Daten setzen wir jedoch ohnehin auf eigene Software, so dass uns dies nicht weiter stört.

3.5 Globaler L2- und L3-Cache

Zur Erleichterung der Fehlersuche, für's Monitoring und für statistische Auswertungen sammeln wir die MAC- und IP-Adressen aus den MAC-Adress-Tabellen der Switche sowie den ARP- und Neighbor-Tabellen aller Router und merken sie uns eine Weile in einer MySQL-Datenbank. Wir haben also quasi eine große Tabelle, in der alle Informationen stehen und müssen zur Informationsfindung nicht unbedingt direkt auf den Geräten suchen.

Alle 20 Minuten startet `mac_collector.pl` und läuft über alle Router und Switche. Dabei wird im Wesentlichen auf jedem Gerät so etwas wie `show mac-address-table` ausgeführt, das Ergebnis interpretiert und mit Timestamp in die Datenbank geschrieben. Heraus kommt dabei eine Zuordnung von MAC-Adressen zu Switchports. Bei der derzeitigen Anzahl an Geräten braucht das Script zwischen 30 und 70 Minuten für einen Durchlauf, je nach Last/Geschwindigkeit der Geräte und Größe der Tabellen. Da das Script länger läuft als der Abstand zwischen zwei Startvorgängen ist, werden die Geräte in alphabetischer Reihenfolge abgefragt. Es startet jedoch nie öfter als dreimal gleichzeitig, d.h. zu Stoßzeiten erfassen wir die Daten teilweise nur alle 40 Minuten.

Alle fünf Minuten startet `arp_collector.pl` und läuft über alle Router. Dort werden auf ähnliche Art und Weise die ARP- und Neighbor-Tabellen ausgelesen. In den ARP-Tabellen stehen IPv4- und in den Neighbor-Tabellen IPv6-Adressen zusammen mit der zugehörigen MAC-Adresse. Dabei erhalten wir eine Zuordnung von IP- bzw. IPv6-Adressen zu MAC-Adressen. Zusammen mit den Daten von `mac_collector.pl` haben wir also eine Zuordnung von IP-Adressen zu Switchports.

`maclookup.pl` findet schließlich eine IP- oder MAC-Adresse im globalen Cache. Das Programm hat zahlreiche Optionen und unterstützt mehrere Ausgabeformate: Text, HTML und Dumper, eine Ausgabe, die direkt mit dem Perl-Modul `Data::Dumper` erzeugt wird. Man kann mit dem Tool entweder nach IP- oder MAC-Adressen suchen und sieht dann, an welchen Ports die Adresse bekannt war oder man sucht nach Switchports und bekommt Informationen darüber, welche IP- und MAC-Adressen dort bekannt waren.

3.6 Hilfsprogramme

Port-Umschaltungen, das Bridgen und Anlegen von VLANs und andere einfache Dinge können wir per Webinterface (HirnMS) bequem zusammen klicken und auch zeitgesteuert ausführen. Insbesondere auf Routern oder bei größeren Umzügen und Neueinrichtungen wird jedoch oft einiges von Hand oder per eigens dafür geschriebenem Script auf den Geräten verändert. Damit diese Änderungen in der Datenbank landen gibt es zwei Scripte, die gegebenenfalls aufgerufen werden müssen:

`vlan-query.pl` liest VLANs von Routern in die Datenbank ein bzw. aktualisiert die Informationen in der Datenbank. Da wir auf Routern ausser simplen Port-Umschaltungen prinzipiell nichts per Automat konfigurieren, ist der Aufruf dieses Tools nach allen VLAN-Änderungen nötig, beispielsweise nach der Zuweisung neuer IP-Subnetze oder dem Umzug eines Layer-3-VLANs auf einen anderen Router.

`syncswitchtodb.pl` wurde weiter oben bereits erwähnt. Es liest von einem Gerät die komplette Port-Konfiguration in die Datenbank ein, unter Berücksichtigung bereits vorhandener Daten.

4 Netzbetreuer-Tools

Für Netzbetreuer stellen wir einige Hilfsmittel zur Verfügung, um sie bei ihrer Arbeit zu unterstützen. Das ist für alle Seiten von Vorteil, denn so können Netzbetreuer viele Aufgaben unverzüglich erledigen und uns bleiben einige E-Mails und Anrufe zu Standardaufgaben erspart. Denn wir müssen bei der Größe unseres Netzes und der zur Verfügung stehenden Arbeitskraft darauf achten, den Betriebsablauf möglichst effizient zu gestalten.

4.1 Alice

Mit *Alice* haben wir ein Web-Frontend für Netzbetreuer geschaffen, unter dem wir Administrations-Tools für Netzbetreuer vereinen und mit dessen Hilfe einige Netzverwaltungsaufgaben bequem im „self service“ erledigt werden können:

- ACL Management
- Abrufen von Informationen (Port-Status, aktive MAC-Adressen) zu Ports
- Umschalten von Ports zwischen den betreuten VLANs
- Schalten von H.I.R.N. Ports in berechtigten Bereichen
- Suche nach MAC- und IP-Adressen
- Weitervergabe von Rechten an weitere Netzbetreuer

Netzbetreuer sehen in einer Baumstruktur gruppiert nach Gebäude, Etage und Raum jeden Raum, in welchem sich mindestens ein Port befindet, der in eines der ihm zugeordneten VLANs geschaltet ist. Es werden dann alle Ports dieses Raums angezeigt, auch H.I.R.N.-Ports und Ports anderer VLANs in entsprechend anonymisierter Form. H.I.R.N.-Ports können per Dropdown-Liste in andere, dem Netzbetreuer zugeordnete VLANs umgeschaltet werden. Ebenso können „eigene“ Ports in H.I.R.N.-Ports umgeschaltet werden.

Die Suche nach MAC- und IP-Adressen funktioniert nur eingeschränkt, das heißt, dass die zu suchende Information sich gerade im ARP-Cache oder in den MAC-Adress-Tabellen der Geräte befinden muss. Es wird „live“ gesucht, das heißt, es kann auch mal ein, zwei Minuten dauern bis ein Ergebnis kommt. Dafür ist die Information dann auch aktuell.

Es steht jedem Netzbetreuer frei, Alice zu nutzen und/oder Dinge bei uns per Mail zu beauftragen. Alice wird jedoch von den Netzbetreuern sehr gut angenommen.

5 Abschliessendes

Viele Details unserer Arbeit und Aspekte des Datennetzes der Ruhr-Universität Bochum gehen aus diesem Artikel nicht hervor. Beispielhaft seien hier die Anbindung der Wohnheime, Einwahlleitungen und VPN genannt.

Auch mit wenig Manpower schaffen wir es, ein verhältnismäßig großes strukturiertes Netz effizient zu verwalten. Dies gelingt uns trotz der Masse der Geräte und der Heterogenität des Netzes. Nicht zuletzt gelingt uns dies jedoch auch wegen der zahlreichen Software-Eigenentwicklungen.

Autorenrichtlinien
Hilfreiches für alle Beteiligten

Selbst etwas für die UpTimes schreiben? Gern. Jedes Thema ist willkommen, das ein GUUG-Mitglied interessiert und im Themenbereich der GUUG liegt. Was sonst noch zu beachten ist, steht in diesen Autorenrichtlinien.

Wir sind an Beiträgen immer interessiert. Wir – das ist diejenige Gruppe innerhalb der GUUG, die dafür sorgt, dass die UpTimes entsteht. Dieser Prozess steht jedem GUUG-Mitglied offen. Der Ort dafür ist die Mailingliste <redaktion@uptimes.de>.

Themen und Zielgruppe

Die UpTimes richtet sich als Vereinszeitschrift der GUUG an Leser, die sich meistens beruflich mit Computernetzwerken, IT-Sicherheit, Unix-Systemadministration und artverwandten Themen auseinandersetzen. Technologische Diskussionen, Methodenbeschreibungen und Einführungen in neue Themen sind für dieses Zielpublikum interessant, Basiswissen im Stil von *Einführung in die Bourne Shell* hingegen eher nicht. Wer sich nicht sicher ist, ob sein Thema für die UpTimes von Interesse ist, kann uns gern eine E-Mail an <redaktion@uptimes.de> schicken.

Länge

Ein einseitiger Artikel hat mit zwei Zwischentiteln um die 2.700 Anschläge. Mit etwa 15.000 Anschlägen – inklusive 3 Abbildungen – landet man auf rund vier Seiten. Wir nehmen gern auch zum Beispiel achtseitige Artikel, achten dabei aber darauf, dass der Zusammenhang erhalten bleibt und dass es genug Bilder gibt, damit keine Textwüsten enstehen. Wer Interesse hat, für die UpTimes zu schreiben, macht sich am besten um die Zeichenzahl nicht so viele Gedanken – auch für kurze oder lange Formate finden wir einen Platz. Die Redaktion ist auch bei der konkreten Ideenentwicklung behilflich. Für eine Artikelidee an <redaktion@uptimes.de> reicht es, wenn Ihr ein bestimmtes Thema behandeln wollt.

Beitragsarten

Neben Fachbeiträgen sind Leserbriefe, Buchrezensionen, Konferenzberichte und Berichte aus dem Vereinsleben für uns immer interessant. Wer sich also nicht gleich traut, mehrseitige Artikel zu schreiben, darf sich gerne erstmal an kleineren Beiträgen versuchen.

Formate

LaTeX: Wir setzen die UpTimes damit. Wer es nicht kennt: Das ist ein Satzsystem, das aus dem technischen Buchdruck kommt. Daher können Autoren uns gern TeX- oder LaTeX-Dateien zusenden. Weil wir – wie es sich beim Publizieren gehört – mehrspaltig setzen und ein homogenes Erscheinungsbild anstreben, verwenden wir für die UpTimes aber bestimmte Formatierungen. Es ist deswegen relativ sinnlos, ausgefeilte Layoutanweisungen oder neben der Datei *Artikel.tex* noch Funktionsdateien einzusenden, die für das heimische Kompilieren nötig waren. Wir behalten uns vor, Texte für die Veröffentlichung in der UpTimes umzuformatieren. Eine Vorlage mit den von uns verwendeten Auszeichnungen etwa für Tabellen, Kästen und Abbildungen gibt es per Anfrage unter <redaktion@uptimes.de>.

ASCII: Alternativ können wir sehr gut blanke ASCII-Texte (UTF-8) verarbeiten.

OpenOffice bzw. LibreOffice, Word, PDF: Alle drei haben eine Formatierung, die wir grundsätzlich nicht übernehmen können.

Bilder: Wir können alle gängigen Bildformate verwenden, soweit ImageMagick sie verdaut und sie einigermaßen hochauflösend sind.

Am besten eignen sich aber PNG- oder PDF-Dateien (für Bilder, nicht für Texte – siehe oben). Plant bei längeren Artikeln mit 1 Abbildung pro 300 Zeichen. Das müssen aber nicht nur Bilder sein, sondern auch Tabellen, Listings oder ein Exkurskasten.

Listings: Der mehrspaltige Druck erlaubt maximal ca. 40 Zeichen Breite für Code-Beispiele. Breitere Listings setzen wir entweder als separate Abbildung, oder wir formatieren um.

Manuskripte einreichen

Am einfachsten ist es, wenn wir ein Manuskript per E-Mail an <redaktion@uptimes.de> erhalten. Das ist jederzeit möglich, spätestens jedoch vier Wochen vor dem Erscheinen der nächsten UpTimes. Zum eigentlichen Manuskript ist ein kleiner Infotext zum Autor wichtig, ein Bild ist sehr wünschenswert.

Wir sind sehr dankbar, wenn der Text vor Einsendung durch eine Rechtschreibkorrektur gelaufen ist. `aspell`, `ispell` oder `flyspell` für Textdateien sowie die von LibreOffice bieten sich an. Wir redigieren zwar die Einsendungen, doch üblicherweise bekommt man erst in mehreren Anläufen einen fehlerfreien Text hin. Wir halten uns an die neue deutsche Rechtschreibung.

Redaktion und Satz

Wir behalten uns vor, Texte für die Veröffentlichung in der UpTimes zu kürzen, und wir redigieren auch etwas. Das bedeutet, dass versehentliche Leeraussagen wegfallen, Syntax und Satzanschlüsse glatter werden, dass Passiva und Substantivierungen verringert oder auch Unklarheiten beseitigt werden (die zum Beispiel Fragen offen lassen oder aus Passivkonstruktionen resultieren, ohne dass der Schreibende das merkt). Manchmal ist dieser Prozess auch mit Nachfragen an den Autoren verbunden.

Die endgültige Textversion geht jedem Autoren am Ende zur Kontrolle zu. Dabei geht es um die inhaltliche Kontrolle, ob sich durch den Redaktionsprozess Fehler eingeschlichen haben – das ist weniger dazu gedacht, alles noch einmal umzuwerfen oder Verschreiber herauszusuchen. Ein bis zwei Wochen vor dem geplanten Erscheinungstermin setzt die Redaktion die Artikel. Nach Redaktion und Satz wird die UpTimes dann zum geplanten Termin veröffentlicht.

Rechtliches

Die Inhalte der UpTimes sollen ab Veröffentlichung unter der CC-BY-SA-Lizenz stehen, damit jeder Leser die Artikel und Bilder bei Nennung der Quelle weiterverbreiten und auch weiterverarbeiten darf. Bei allen eingereichten Manuskripten gehen wir davon aus, dass der Autor sie selbst geschrieben hat und der UpTimes ein nicht-exklusives, aber zeitlich und räumlich unbegrenztes Nutzungs- und Bearbeitungsrecht unter der CC-BY-SA einräumt.

Bei Fotos oder Abbildungen, die nicht selbst erstellt wurden, flutscht das gern mal durch: Es ist wichtig, dass der Autor sich bei dem Urheber die Erlaubnis zu dieser Nutzung einholt, und fragt, wie die Quelle genannt zu werden wünscht. Die Frage nach der CC-BY-SA ist hierbei besonders wichtig.

An Exklusivrechten, wie sie bei kommerziellen Fachzeitschriften üblich sind, hat die UpTimes kein Interesse. Es ist den Autoren freigestellt, ihre Artikel noch anderweitig nach Belieben zu veröffentlichen.

Finanzielles

Für Fachbeiträge, die einen eigenen inhaltlichen Anspruch erheben, zahlt die GUUG dem Autor nach Rechnungstellung durch den Autor pro Seite 50 € zuzüglich eventuell anfallender USt. Leserbriefe, Buchrezensionen und Artikel bezahlter Redakteure sind davon ausgenommen. Gleiches gilt für Paper, wenn die UpTimes die Proceedings der Konferenz beinhaltet.

Nächste Ausgabe (digital):
UpTimes 02-2013, Sommer-Ausgabe

- Redaktionsschluss: Sonntag, 2. Juni 2013.
- Erscheinung: 31. Juli 2013.
- Gesuchte Inhalte: Fachbeiträge über Unix und verwandte Themen; Veranstaltungsberichte; eigene Buchvorstellungen, Rezensionen; Beiträge zum Vereinsleben.
- Artikelideen und Manuskripte an: <kehrer@guug.de> oder direkt an die Mailingliste <redaktion@uptimes.de>

Über die GUUG

Vereinigung deutscher UNIX-Benutzer

German Unix User Group e.V.

Die Vereinigung Deutscher Unix-Benutzer hat gegenwärtig 678 Mitglieder, davon 86 Firmen und Institutionen.

Im Mittelpunkt der Aktivitäten der GUUG stehen Konferenzen. Ein großes viertägiges Event der GUUG hat eine besondere Tradition und fachliche Bedeutung: In der ersten Jahreshälfte treffen sich diejenigen, die ihren beruflichen Schwerpunkt im Bereich der IT-Sicherheit, der System- oder Netzwerkadministration haben, beim *GUUG-Frühjahrsfachgespräch* (FFG).

Seit Oktober 2002 erscheint mit der *UpTimes* – die Sie gerade lesen – eine Vereinszeitung. Seit 2012 erscheint die UpTimes einerseits zu jedem FFG in Form einer gedruckten Proceedings-Ausgabe (ISBN), und andererseits im Rest des Jahres als digitale Redaktionsausgabe (ISSN). Daneben erhalten GUUG-Mitglieder zur Zeit die Zeitschrift *LANline* aus dem Konradin-Verlag kostenlos im Rahmen ihrer Mitgliedschaft.

Schließlich gibt es noch eine Reihe regionaler Treffen (http://www.guug.de/lokal): im Rhein-Ruhr- und im Rhein-Main-Gebiet sowie in Berlin, Hamburg, Karlsruhe und München.

Warum GUUG-Mitglied werden?

Die GUUG setzt sich für eine lebendige und professionelle Weiterentwicklung im Open Source-Bereich und für alle Belange der System-, Netzwerkadministration und IT-Sicherheit ein. Wir freuen uns besonders über diejenigen, die bereit sind, sich aktiv in der GUUG zu engagieren. Da die Mitgliedschaft mit jährlichen Kosten

Fördermitglied	350 €
persönliches Mitglied	90 €
in der Ausbildung	30 €

verbunden ist, stellt sich die Frage, welche Vorteile damit verbunden sind?

Neben der Unterstützung der erwähnten Ziele der GUUG profitieren Mitglieder auch finanziell davon, insbesondere durch die ermäßigten Gebühren bei den Konferenzen der GUUG und denen anderer europäischer UUGs. Mitglieder bekommen außerdem *c't* und *iX* zum reduzierten Abopreis.

Wie GUUG-Mitglied werden?

Füllen Sie einfach das umseitige Anmeldeformular aus und schicken Sie es per Fax oder Post an die unten auf dem Formular angegebene Adresse. Falls Sie die Seite nicht herausreißen wollen: Sie können den Mitgliedsantrag als PDF herunterladen, siehe URL auf dem Mitgliedsantrag.

Impressum

UpTimes – Mitgliederzeitschrift der German Unix User Group (GUUG) e.V.
Herausgeber: GUUG e.V.
Bruno-Walter-Ring 30
D-81927 München
Tel.: +49-(89)-380 125 95 0
Fax: +49-(89)-380 125 95 9
E-Mail: <redaktion@uptimes.de>
Internet: http://www.guug.de/uptimes/

Autoren dieser Ausgabe: Christian Bockermann, Volker A. Brandt, Marc Haber, Robin Schröder, Karsten Schulz, Dr. Udo Seidel
V.i.S.d.P: Wolfgang Stief, Vorstandvorsitzender, Anschrift siehe Herausgeber
Chefredaktion: Robin Schröder
LaTeX-Layout (PDF): Robin Schröder
Titelbild und -gestaltung: Hella Breitkopf
Druck: Druck Consulting Kuhnert Schablow Vertriebs-GmbH, Mühlenstr. 22a, 28832 Achim
Verlag: Lehmanns Media GmbH, Hardenbergstraße 5, 10623 Berlin
ISSN: 1860-7683
ISBN: 978-3-86541-489-2

German Unix User Group e.V.
Mitgliedsantrag

Name, Vorname: .....

Firma/Organisation: .....

.....

Straße: .....

PLZ, Ort: .....

E-Mail: .....

Ich möchte/wir möchten

☐ persönliches Mitglied (Jahresbeitrag 90 €)
  ☐ Ich bin in der Ausbildung (bitte Nachweis beifügen) und zahle nur 30 €
☐ Fördermitglied (Jahresbeitrag 350 €)

werden.

☐ Bitte richten Sie mir den Alias ..... @guug.de ein.

☐ Ich bin mit der Weitergabe meiner Adressdaten an Verlage, die GUUG-Mitglieder im Rahmen der Mitgliedschaft kostenlos mit Zeitschriften beliefern, einverstanden.

Ich bin damit einverstanden, dass die o. g. Daten durch die GUUG elektronisch gespeichert werden. Mir ist bekannt, dass die Kündigung der Mitgliedschaft nur mit einer Frist von drei Monaten zum Ende des Kalenderjahres möglich ist.
Die Satzung ist unter http://www.guug.de/verein/satzung/ einsehbar.

....
Ort, Datum Unterschrift

Ich ermächtige die GUUG bis auf Widerruf zum jährlichen Einzug des Mitgliedsbeitrags von meinem Konto:

Name des Kontoinhabers: .....

Kontonummer: .....

BLZ: .....

Bank: .....

....
Ort, Datum Unterschrift

Bitte faxen Sie diesen Antrag an +49-(89)-380 125 95 9 oder schicken ihn an:
GUUG e.V. * Postfach 25 01 23 * D-44739 Bochum

Dieses Formular ist auch unter http://www.guug.de/verein/mitglied/guug-anmeldeformular.pdf zu finden.