Proceedings des Deutschen Perl Workshops

WS-Orga-Team (Hrsg.)

Ort / Location
Datum / Date

Inhaltsverzeichnis

1 Dummy-Artikel 1

Autor

Sabine Mustermann

Abstract

Mustertext

2 RADIUS in Perl: Radiator

2.1 Autor

Kurt Jaeger (<pi@nepustil.net>),

2.2 Bio Kurt Jaeger

Kurt Jaeger ist Geschäftsführer eines regionalen ISPs (Dr.-Ing. Nepustil & Co. Gmbh), im Vorstand der ISP Service eG, einer Einkaufsgenossenschaft von Internet Service Providern, und Perl Anwender seit 1992.
Er nutzt seit 1998 die RADIUS Server Software Radiator.

2.3 Abstract

Praktisch alle grossen TK-Netze (DSL-Plattformen) realisieren u.a. auch Zugangskontrolle und Abrechnung der Zugänge. Diese Prozesse finden sehr oft mit Hilfe des RADIUS Protokolls statt (RFC2865).
`http://www.open.com.au/radiator/index.html`: Radiator, eine kommerzielle Perl Anwendung wird ausführlich vorgestellt.
Die Anwendung ist seit über 10 Jahren weltweit auch in den grössten Netzen verbreitet. Den meisten Anwendern ist völlig unbekannt, daß eine solche Anwendung in Perl realisiert wird und was sie alles leistet.
Der Artikel gibt eine Übersicht über RADIUS, die Implementierung in Radiator und schildert Beispielanwendungen.

2.4 Was ist RADIUS ?

RADIUS ist ein von der IETF 1997 erstmals standardisiertes Protokoll für die Benutzer-Authentisierung. Eine Benutzerin, die eine Netzresource verwenden möchte, wird von einem Einwahlrouter (Network Access Server, NAS) nach einem Namen und einem Passwort gefragt, und der Router fragt per RADIUS-Protokoll nach, ob Zugang gewährt werden soll.
Aktuell beschreibt `http://www.ietf.org/rfc/rfc2865.txt`: RFC2865
das Protokoll, mit ergänzenden Hinweisen in `http://www.ietf.org/rfc/rfc5080.txt`: RFC5080.
Das RADIUS Protokoll wird mit UDP Paketen implementiert.
Durch IANA sind die Ports 1812 für die Authentisierung und 1813 für Abrechnungsdaten `http://www.iana.org/assignments/port-numbers`: vorgesehen. In älteren Installationen sind noch Ports 1645 und 1646 in Verwendung.
RADIUS hat seinen Vorläufer in `http://de.wikipedia.org/wiki/TACACS`: TACACS+

und einen seit einigen Jahren in Diskussion befindlichen Nachfolger in `http://www.opendiameter.org/`: DIAMETER.
Alle drei Protokolle decken die Anwendungsfälle von Authentisierung, Authorisierung und Abrechnung ab, bekannt unter dem Kürzel AAA. Die Authentisierung klärt, wer mit wem spricht, Authorisierung definiert, welche Resourcen dem identifizierten Kommunikationspartner bereitgestellt werden (und welche nicht) und unter Abrechnung versteht man die Aufzeichnung der Menge an verbrauchten Resourcen.

2.5 Beispiele von Network Access Servern (NAS)

Ursprünglich wurden Einwahlrouter mit Modem oder Seriellschnittstellen als Network Access Server bezeichnet. Hersteller und Modelle waren beispielsweise Livingston Portmaster, Ascend MAX 4000 oder Cisco 5200.
Diese Geräte haben und hatten nur geringe lokale Speicherkapazitäten und speichern daher selbst keine Benutzer-Kennungen, sondern verwenden RADIUS.
Im Windows Umfeld ist auch der Begriff Remote Access Server (RAS) in Verwendung. Vorteil dieses Begriffs ist, dass er nicht mit Network Attached Storage verwechselt wird.
Seit dem Aufkommen vieler weiterer Technologien haben auch andere Geräte NAS-Funktionen:

- WLAN Access Points
- Ethernet Switche
- Unix Hosts (PAM!)
- Broadband Remote Access Server (BBRAS) Mit diesen Geräten sprechen DSL Router bei der Einwahl ins Internet.
- uvam

Diese NAS erledigen nach erfolgreicher Authentisierung eine Reihe von Aufgaben, u.a. die Zuweisung von IP-Adressen und weiteren Routen. Eine weitere Aufgabe kann das Setzen von Filtern sein: wenn beispielsweise ein Benutzer-PC als virenverseucht erkannt wurde, wird bei der nächsten Einwahl ins Netz automatisch eine Menge von Filtern angewendet, so dass der PC nur noch bis zur nächsten Antiviren-Webseite kommt.
Weitere Parameter, die bei der Einwahl gesetzt werden können, sind Quality-of-Service-Parameter, Bandbreitenlimits oder automatische Tunnelverbindungen zu bestimmten Netzknoten oder eine spezielle Verschlüsselung.

2.6 Wie funktioniert RADIUS ?

RADIUS ist wie viele Protokolle eine Client/Server Anwendung, bei der NAS Systeme die RADIUS-Clients darstellen und die Hosts, auf denen die Kennungsdaten vorgehalten werden, die RADIUS Server darstellen.
Eine Anfrage besteht aus einem UDP Paket als RADIUS Request und einem Antwortpaket als RADIUS Response. Ist die Infrastruktur komplexer, kann auch der Paketfluss ein wenig komplexer werden.
Die per UDP ausgetauschten Daten sind jedoch meist sehr einfach strukturiert. Die UDP-Pakete sind daher auch sehr einfach parsebar, und mit aktuellen tcpdump-Versionen einfach zu betrachten:

```
auth# tcpdump -i fxp0 -n -s 1500 -vv host <radiushost> and port 1812
[...]
15:15:20.639322 <nasip>.1645 > <radiushost>.1812: [udp sum ok]
  rad-access-req 190 [id 79] Attr[  Framed_proto{PPP} User{username}
  Pass NAS_port{192} NAS_port_id{Uniq-Sess-ID192}
  Calling_station{ffff-ffff-ffff eth 7/0/0:4096.4096 0/0/0/0/0/0}
  Connect_info{1184000/160000} NAS_port_type{Virtual} Service_type{Framed}
  NAS_ipaddr{<nasip>} NAS_id{nashostname} ] (ttl 254, id 9491, len 218)
```

Ursprünglich wurden RADIUS Zugangsdaten wie Benutzername und Passworte aus einfachen ASCII-Dateien gelesen, der Begriff "RADIUS users file" stammt daher. Eine Benutzerdefinition ist also der Ausschnitt aus einem RADIUS users file, der einer Kennung entspricht.

2.7 Beispiel 1: RADIUS Benutzereintrag für Telnet

```
username          User-Password = "password"
    Service-Type = Login-User,
    Login-Service = Telnet,
    Login-IP-Host = 192.168.10.1,
    Ascend-Idle-Limit = 600
```

Bedeutung: Wenn Benutzer 'username' sich mit 'password' anmeldet, dann wird er per Telnet-Protokoll zum Host 192.168.10.1 verbunden. Wenn er 600 Sekunden nichts tippt, trenne die Verbindung.

2.8 Beispiel 2: RADIUS Benutzereintrag für PPP

```
username@my-dsl      User-Password = "password"
```

```
    Service-Type = Framed-User,
    Framed-Protocol = PPP,
    Framed-MTU = 1456,
    Framed-IP-Address = 192.168.5.1,
    Framed-IP-Netmask = 255.255.255.255,
    Framed-Compression = Van-Jacobsen-TCP-IP
```

Bedeutung: Wenn Benutzer 'username' sich mit 'password' anmeldet, baue eine PPP-Verbindung auf, weise ihm IP-Adresse 192.168.5.1 zu und die MTU 1456.

2.9 Beispiel 3: RADIUS Benutzereintrag für L2TP Tunnel

```
username
    Service-Type = Outbound-User,
    cisco-avpair = "vpdn:ip-addresses=10.1.1.1",
    cisco-avpair = "vpdn:tunnel-id=mytunnelid",
    cisco-avpair = "vpdn:tunnel-type=l2tp",
    cisco-avpair = "vpdn:l2tp-tunnel-password=tunnelpw",
    cisco-avpair = "vpdn:source-ip=192.168.1.1"
```

Bedeutung: Wenn eine Verbindung hereinkommt mit User "username", baue eine L2TP Verbindung zu 10.1.1.1 auf, nimm als Quell-IP die 192.168.1.1, und verwende diese Tunnel-ID und dieses Tunnel-Passwort. In der Benutzerdefinition ist kein Passwort angegeben, also wird auch kein Passwort überprüft! Das macht hoffentlich der Zielhost 10.1.1.1.

2.10 Allgemeiner: RADIUS Request/Response

Ein NAS sendet im RADIUS Request also im Allgemeinen eine Menge an Elementen, und erhält als Response wiederrum eine Menge an Elementen.
Jedes Element ist eindeutig bezeichnet durch Typ, Länge und Wert (value), und wird daher auch als 'TLV-kodiert' bezeichnet.
Jedem Request und Response wird noch eine Id (8bit) mitgegeben, so dass Responses den offenen Requests eindeutig zugeordnet werden können.
Schickt ein NAS mehrere Requests hintereinander los, erfolgen die Antworten nicht notwendigerweise in derselben Reihenfolge. Daher ist die Id zwingend.
Jedem Request und Response beigefügt ist ein Authenticator, der durch ein zwischen Client und Server vereinbarten 'shared secret' und der Anwendung von ein wenig Crypto Replay-Attacken unwahrscheinlich macht.

2.11 Welche Elemente sind möglich ?

Ca. 120 Standard-Elemente sind bei IANA `http://www.iana.org/assignments/radius-types`: verzeichnet.
Dazu kommen ca. 180 Elemente, die in den Anfangszeiten des Protokolls Verwendung fanden und noch heute finden.
Dazu gibt es ca. 2000 bekannte herstellerspezifische Elemente (vendor-specific attributes, VSA). Die herstellerspezifischen Attribute werden durch von IANA verwaltete Nummern (`http://www.iana.org/assignments/enterprise-numbers`: enterprise numbers) differenziert.
Die Elemente sind in den RADIUS Servern nicht fest eincompiliert, sondern werden zur Laufzeit aus einer kleinen Menge an zugelassenen Datentypen zusammengebaut.
Die Konfigurationsdatei, in der die dem RADIUS Server bekannten Elemente stehen, ist reiner Text und wird als RADIUS dictionary bezeichnet.
Hier ein Ausschnitt aus dem ca. 6000 Zeilen langen dictionary, welches dem Radiator beiliegt.

```
ATTRIBUTE Service-Type    6            integer
VALUE       Service-Type  Login-User             1
VALUE       Service-Type  Framed-User            2
VALUE       Service-Type  Callback-Login-User    3
[...]
VALUE       Service-Type  Fax                    13
```

2.12 Was kann man bei RADIUS Accounting sehen ?

RADIUS Accounting Records sehen weitgehend wie andere RADIUS Pakete, daher: Eine Menge an Elementen, TLV-kodiert. Diese Daten können ohne grossen Aufwand als reine Textdateien gespeichert werden und später Abrechnungsprozessen zugeführt werden.
Für die Abrechnung werden zu Beginn einer Verbindung Start-Records erzeugt, und am Ende einer Verbindung sogenannte Stop-Records. Zum Status-Update einer Verbindung sind zwischendurch weitere Records erlaubt (Interim-Update), die in der Praxis aber wenig Verwendung finden.
Alle relevanten Angaben sind eigentlich in den Stop-Records enthalten, so dass Start-Records nicht wirklich notwendig sind.
Für die Fehlersuche während einer bestehenden Verbindung ist die Auswertung des Start-Records aber hilfreich, falls die IP-Adresse dynamisch zugewiesen wurde: Darin ist festgehalten, welche IP-Adresse zugewiesen wurde.
Es folgt ein sehr ausführliches Beispiel:

```
Sun Oct 26 20:40:32 2008
    Acct-Session-Id = "00000130"
    Tunnel-Type:0 = L2TP
    Tunnel-Medium-Type:0 = IPv4
    Tunnel-Server-Endpoint:0 = "192.168.1.1"
    Tunnel-Client-Endpoint:0 = "10.1.1.1"
    Tunnel-Assignment-Id:0 = "ISPEG"
    Tunnel-Client-Auth-Id:0 = "lac_srv_clnt"
    Tunnel-Server-Auth-Id:0 = "LNS-SRV"
    Acct-Tunnel-Connection = "436601532"
    Framed-Protocol = PPP
    Framed-IP-Address = 192.168.10.1
    Framed-IPv6-Route = "2001:14f8:0700:0008::/64"
    Framed-IPv6-Route = "2001:14f8:0400::/48"
    Framed-Interface-Id = 0:0:0:1
    User-Name = "testing@my-dsl"
    Acct-Authentic = RADIUS
    X-Ascend-Connect-Progress = LAN-Session-Up
    Cisco-AVPair = "connect-progress=LAN Ses Up"
    X-Ascend-PreSession-Time = 0
    X-Ascend-Xmit-Rate = 17696000
    Cisco-AVPair = "nas-tx-speed=17696000"
    X-Ascend-Data-Rate = 1184000
    Cisco-AVPair = "nas-rx-speed=1184000"
    Acct-Session-Time = 342
    Acct-Input-Octets = 1102
    Acct-Output-Octets = 1667
    X-Ascend-Pre-Input-Octets = 0
    X-Ascend-Pre-Output-Octets = 40
    Acct-Input-Packets = 51
    Acct-Output-Packets = 57
    X-Ascend-Pre-Input-Packets = 0
    X-Ascend-Pre-Output-Packets = 2
    Acct-Terminate-Cause = User-Request
    X-Ascend-Disconnect-Cause = PPP-Rcv-Terminate-Req
    Cisco-AVPair = "disc-cause-ext=PPP Receive Term"
    Acct-Status-Type = Stop
    NAS-Port = 105
    NAS-Port-Id = "Uniq-Sess-ID105"
    Connect-Info = "17696000/1184000"
    NAS-Port-Type = Virtual
```

```
Service-Type = Framed-User
NAS-IP-Address = 10.1.1.1
NAS-Identifier = "dsl-gw.my-dsl"
Acct-Delay-Time = 0
Client-IP-Address = 10.1.1.1
Acct-Unique-Session-Id = "d03978a8140211a7"
Timestamp = 1225050032
```

2.13 Radiator

Radiator ist eine Implementierung eines RADIUS Servers. Ca. 8000 Installationen in 180 Ländern deuten auf die weite Verbreitung hin. Die lizenzpflichtige Software ist zu 100% in Perl geschrieben und wird auf dem Zielsystem wie normale andere Perl-Module installiert.
Die Entwicklerfirma, Open Systems Consultants in Australien, verlangt für eine Single-Server Lizenz ca. 1000 australische Dollar, das sind mit heutigem Wechselkurs ca. 600 EUR.
Die Software läuft auf praktisch allen Plattformen, auf denen Perl verfügbar ist. Da der Source-Code ausgeliefert wird und exzellent kommentiert und dokumentiert ist, ist eine Anpassung jederzeit und einfach möglich.
Die Dokumentation ist unter `http://www.open.com.au/radiator/ref.pdf` erhältlich.

2.14 Programm-Start

Das Programm wird mit folgendem Aufruf gestartet:

```
/usr/local/bin/radiusd -daemon -config_file /usr/local/etc/radiator/conf
```

Der radiusd liest das Config-File, startet bei Bedarf einige Initialisierungsschritte wie Datenbankverbindungen und wartet dann auf den RADIUS Standard Ports auf einkommende Requests.
Es werden keine weiteren Subprozesse des radiusd gestartet und der Prozess muss für keine Logfile-Transaktion neu gestartet werden. Der Prozess läuft ohne Threads.
Hier eine Beispielkonfigurationsdatei:

```
# --------------------
Foreground
Trace           5
AuthPort        1812
AcctPort        1813
LogDir          /var/log/radiator
```

```
DbDir           /usr/local/etc/radiator
DictionaryFile /usr/local/etc/radiator/dictionary
PidFile         /var/log/radiator/pid
LogFile         /var/log/radiator/log.%Y%m%d
BindAddress     192.168.7.1,ipv6:2001:14f8:0200:0000:0000:0000:0000:000f

# NAS Devices (Clients)
<Client ipv6:2001:14f8:0200:0000:0000:0000:0000:000f>
    Secret ArAtVap1
    DupInterval 0
    IgnoreAcctSignature
</Client>

<SNMPAgent>
    Port 1814
    ROCommunity Ejukhacji
    BindAddress 213.178.180.15
</SNMPAgent>

<AuthLog FILE>
    Identifier flat
    Filename /var/log/radiator/flat
    SuccessFormat %l:%u:%P:OK
    FailureFormat %l:%u:%P:FAIL
    LogSuccess 1
    LogFailure 1
</AuthLog>

<Handler>
    AuthLog flat
    AcctLogFileName /var/log/radiator/flat.%Y%m%d
    AuthByPolicy ContinueWhileReject
    <AuthBy FILE>
        Filename /usr/local/etc/radiator/users/flatfile
    </AuthBy>
</Handler>

# --------------------
```

2.15 Verarbeitung von Requests

Eingehende Requests werden anhand der Absender-IP-Adresse dem jeweiligen Client zugeordnet und dekodiert. Je nach enthaltenen Elementen wird ein passender Handler ausgewählt, der die Anfrage bearbeiten wird. In jedem Handler ist eine Folge von Authentisierungsmodulen definiert, die in der im config-File festgelegten Reihenfolge durchprobiert werden.
Eine Mischung unterschiedlicher Handler und Authentisierungsmodule ist möglich, um verschiedene Anforderungen abzudecken. Hier Beispiele dazu:

```
<Handler User-Name=/\@my-dsl$/,Request-Type=Access-Request>
    AuthLog my-dsl-authlogfile
    AuthByPolicy ContinueWhileReject
    <AuthBy FILE>
        Filename /usr/local/etc/radiator/users/my-dsl-users
    </AuthBy>
    <AuthBy RADIUS>
        <Host 192.168.1.15>
            Secret Roidmied
            AuthPort 1645
            AcctPort 1646
            RetryTimeout 5
        </Host>
    </AuthBy>
</Handler>

<Handler User-Name=/\@my-dsl$/,Request-Type=Accounting-Request>
    AuthByPolicy ContinueUntilReject
    AcctLogFileName /var/log/radiator/details.%Y%m%d
    AccountingHandled
    <AuthBy RADIUS>
        <Host 192.168.1.15>
            Secret Roidmied
            AuthPort 1645
            AcctPort 1646
            RetryTimeout 5
        </Host>
    </AuthBy>
</Handler>
```

Der erste Handler sucht für Access-Requests Kennungen entweder lokal in einer Datei oder von einem anderen RADIUS-Server.
Der zweite Handler leitet für Abrechnungszwecke die eingehenden Records weiter, behält aber eine lokale Kopie.
Bei Radiator sind eine große Zahl an Authentisierungsmodulen bereits vorhanden, die alle vorstellbaren Datenquellen berücksichtigen. Die Benutzerdaten können auch aus Datenbanken abgefragt werden. Es werden MySQL, Oracle, Sybase, Postgres, ODBC, Interbase, Informix, CSV-Files und SQLite2 unterstützt.
Die Abfrage von LDAP-Datenbanken oder Windows Authentisierungen aus Active Directory oder anderen Verfahren ist möglich. Eine übersandte Kennung kann gegen IMAP, POP oder HTTP Server geprüft und ggf. zugelassen werden. Diverse One-Time Passwort Systeme wie DIGIPASS oder RSA SecureID werden unterstützt. Zugriffe auf die Kundenbestandsdaten diverser ISP Provisionierungssysteme sind ebenfalls möglich. Kerberos als Authentisierungsdatenquelle ist unterstützt, und wurde von einem Kunde beigesteuert.
Die Verarbeitung von TACACS+ oder Diameter Requests ist ebenfalls möglich. Diese werden intern dann als RADIUS Requests behandelt, so dass sich die Komplexität für den Systembetrieb in beherrschbaren Grenzen hält.
Die Flexibilität ist so gross, dass sich die Frage nach dem Vergleich mit LDAP, Kerberos, NIS, PAM und ActiveDirectory stellt: Alle diese Systeme zu integrieren geht am Einfachsten vermutlich über RADIUS via Radiator.

2.16 Perl-Code

Der Code umfasst ca. 60K LoC. Der Server selbst benötigt nur ca. 600 Zeilen, der Rest wird über sauber modularisierte Schnittstellen bereitgestellt oder aus CPAN Modulen hinzugefügt.
Das Class Inheritance Diagramm ist in der Dokumentation auf Seite 315 abgedruckt und kann als erste Orientierung dienen.
Da der Quell-Code des Radiator selbst auf Lizenznehmer beschränkt ist, werde ich hier keine Ausschnitte einfügen, sondern mich mit einem Hinweis auf interessante Abschnitte begnügen und im Vortrag Näheres besprechen:

- Radius/Radius.pm
- Radius/ServerHTTP.pm: Einfacher Webserver
- Radius/Resolver.pm: DNS-Lookup
- Radius/SNMPAgent.pm: SNMP Server
- Radius/ApplePasswordServer.pm
- Radius/RSAAM.pm: SecureID

- goodies/nntp-redirect.pl: NNTP Auth für Newsserver

2.17 Beispiele

Einige Anwendungsbeispiele können die Flexibilität des RADIUS Protokolls und der Implementierung als Radiator plastisch vermitteln.

2.18 ISP Service eG

Die ISP Service eG als Einkaufsgemeinschaft regionaler Internet Dienstleister (ISP) hat Zusammenschaltungen mit diversen Carriern für die Übergabe von DSL Verkehr aus verschiedenen Infrastrukturen.

Einkommende DSL Sessions werden entweder lokal authentisiert oder dem jeweiligen Mitglied zur Freigabe per RADIUS Peering weitervermittelt. Die Weiterleitung an den richtigen ISP erfolgt durch Parsen der einkommenden Benutzernamen mit Hilfe regulärer Ausdrücke im Radiator.

Die Sessions können entweder am Zugangsrouter der Genossenschaft oder dem Mitglieds-ISP per L2TP übergeben werden.

Der Genosse kann daher seine Kunde parallel über verschiedene DSL-Infrastrukturen (Telekom, QSC, LambdaNet) Sessions vermitteln und bei Ausfällen kurzfristig durch Angabe einer anderen Kennung den Vorleistungserbringer wechseln, ohne dass sich die zugeteilten IP-Adressen ändern müssen. Je nach DSL Endgerät kann das sogar automatisch erfolgen.

Die Fehlersuche und anschliessende Diskussion mit DSL-Plattform Betreibern ist meist sehr viel einfacher, wenn man auf die bessere Stabilität des Mitbewerbs hinweisen kann, die sich in einfachen MRTG Graphen nachweisen lässt.

Die Abrechnung der Nutzung kann der Mitglieds-ISP anhand der in Realtime zugeführten RADIUS Accounting Records für seine Kunden zeitnah realisieren. Die Prüfung der Abrechnung ist für beide Seiten (ISP Service eG und Mitglied) auf den Abgleich der RADIUS Accounting Records beschränkt. Missbrauchsmöglichkeiten sind durch die Architektur nur mit sehr hohem Aufwand möglich, da beiden Parteien dieselben Abrechnungsdaten als eigene Kopien vorliegen.

Ein Monitoring der RADIUS-Server der Mitglieder ist einfach möglich, aber bis jetzt mangels Dringlichkeit nicht vorhanden.

Redundanz bei Ausfällen ist prinzipiell möglich, aber bisher nicht wirklich dringlich gewesen.

Der RADIUS Server ist eine von vielen Anwendungen auf einer Intel P4 Maschine mit 3 GHz und 1 GB RAM seit vielen Jahren stabil betreibbar.

2.19 EAP: Extensible Authentication Protocol

EAP als Authentisierungsprotokoll ist in `http://www.ietf.org/rfc/rfc2869.txt`: RFC2860
seit Juni 2000 über RADIUS verwendbar.
Managebare Ethernet Switche oder WLAN Accesspunkte erlauben mit Hilfe dieses Protokoll die Kontrolle des Zugangs zum Netz.
In der Dokumentation auf Seite 326ff wird das Verfahren ausführlich dargelegt.
Auf die Demonstration während des Vortrags wird verwiesen.

2.20 EDUROAM

Ein weltweite Zusammenschaltung für den Zugang zu LANs von Hochschulen und Bildungseinrichtungen über EAP und RADIUS Peering ist der EDUROAM Verbund (`http://www.eduroam.org/`, `http://www.eduroam.edu.au/`).
In Europa nehmen 33 Länder an diesem Verbund teil, in Asien 6, darunter China und Japan.
Das Handbuch für die Einrichtung ist online abrufbar und behandelt u.a. die Einrichtung des Radiator.

2.21 NNTP Proxy

Als Teil der Goodies des Radiators ist ein in Perl realisierter NNTP Proxy beigefügt, der das NNTP Authentisierungsprotokoll mit einem NNTP Client spricht, per RADIUS die Berechtigung für den Zugriff abfragt und dann an einen konfigurierbaren NNTP Server alle weiteren Schritte des NNTP Protokolls weiterleitet:

`goodies/nntp-redirect.pl`

2.22 Mobilfunk-Anwendung

Ein von T-Systems realisiertes System ermöglicht 600000 Benutzern aus drei Mobilfunknetzen den Zugriff auf bestimmte Intranet-Inhalte.
Dazu kommen RADIUS-Requests mit den Rufnummern an und werden mit den passenden zugewiesenen IP-Adressen beantwortet.
Im Peak werden 180 RADIUS-Requests pro Sekunde verarbeitet. Die Daten dazu kommen aus einer Oracle-Datenbank.
Die Ausfallsicherheit wird mit zwei RADIUS Proxies realisiert, die an vier RADIUS-Server weitervermitteln, die wiederrum die Datenbank abfragen.
Die sechs Maschinen sind vom Typ Sun x4200, und mit 8 GB RAM, DualCore und Betriebssystem Debian Linux nicht übermässig aufwendig.

2.23 IPv6

Das RADIUS Protokoll ist eine Schlüsselanwendung für grosse Netze. Daher stellt sich schon heute die Frage, ob Radiator bereits heute für den Betrieb in einer gemischten oder reinen IPv6 Umgebung vorbereitet ist.
Alle Kommunikation kann gemischt oder exklusiv auf IPv6 betrieben werden.
Viele NAS-Hersteller sind jedoch noch nicht soweit, dass sie RADIUS bereits ueber IPv6 sprechen. Die Anwendungsebene in Form von RADIUS Attributen ist oft schon vorhanden.

2.24 RADIUS Probleme und Lösungen

1. RADIUS Requests enthalten mit 8bit IDs nur einen sehr beschränkten Nummernbereich, um viele offene Requests gleichzeitig zu bearbeiten. In der Praxis kann das über geeignete Loadbalancing Verfahren und mehrere parallel betriebene RADIUS Server abgefangen werden.

2. Verzögerungen bei der Beantwortung aufgrund externer Datenquellen können sich in komplexen RADIUS Proxy Hierarchien unangenehm aufschaukeln. Die grossen DSL Aggrationsrouter (BBRAS) können gleichzeitig 200-400K DSL Verbindungen halten. Nach Ausfällen grosser Netzbereiche (z.B. Stromausfall) und Wiederanlauf treten kritische Lastspitzen auf, die zu kritischen Verzögerungen bei der Wiederanmeldung am DSL-Netz führen können.

3. Die per RADIUS übertragenen TLVs werden, wie anhand des tcpdumps sichtbar, nicht verschlüsselt. Die Autoren von Radiator haben dafür das RADSEC Protokoll als Lösung vorgeschlagen. Siehe Seite 235 der Dokumentation.

4. RADIUS Requests sind zustandslos. Die Simulation eines sitzungsorientierten Verzeichnisdienstes, z.B. Anzeigen der Benutzernamen via PAM via RADIUS in einem Verzeichnis unter UNIX erzeugt pro Benutzer Radius-Lookups und ist damit nicht praktisch verwendbar.

5. Beim Crash eines NAS werden alle offenen Sessions abgebrochen. Da RADIUS zustandslos ist, verliert man für die Abrechnung möglicherweise viele wichtige Daten.

2.25 Radiator Probleme und Lösungen

1. Radiator arbeitet die eingehenden Requests seriell ab, dies kann bei Lastspitzen zu Aufschaukeleffekten führen, wenn Clients nach kurzem Timeout wiederholt Requests versenden. Die Verzögerungen bei der Beantwortung von Requests sind in der Praxis immer auf die Verwendung externer Datenquellen zurückzuführen und durch geeignetes Loadbalancing oder Caching lösbar.

Keine reale Installation ist je durch die Performance des Radiator oder der Sprache Perl eingeschränkt gewesen.

2.26 Bücher zum Thema RADIUS

1. Deploying Radius `http://deployingradius.com/blog/`.
2. SIP/VoIP und Radius `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470011947.html`.
3. RADIUS `http://oreilly.com/catalog/9780596003227/`.

2.27 Danksagung

Hugh Irvine und Mike McCauley möchte ich für eine sehr Software und Dokumentation gratulieren.
Stefan Feurle von T-Systems möchte ich für den Hinweis auf deren Mobilfunkanwendung danken.

3 Dummy-Artikel 2

Autor

Sabine Mustermann

Abstract

Mustertext