Migration von Applikationen zu IPv6

Hintergrund

- Projekt Deutschland Online Infrastruktur will IPv6 für das Kommunikationsnetz der Behörden
- Absicherung des Netzes nötig
- Ziel:
  - minimale Implementation bis 31.12.09
  - brauchbare Implementation in GeNUGate 7.0

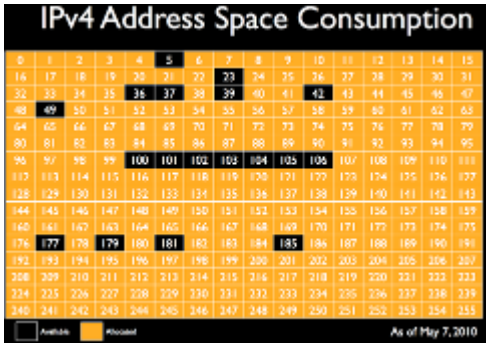
Was ist GeNUGate?

- zweistufige Hochsicherheitsfirewall: Behörden, Banken etc
- basiert auf gehärtetem OpenBSD
- kein Forwarding, sondern Application Level Gateways (ALG):
  - www, smtp, ftp, telnet, pop, nntp...
  - tcp, udp, rawip, ping
- ALGs sind primär in perl geschrieben

Realisierungszeitraum

- erste tiefere Gedanken zum Thema Mitte 2008
- Anfangen Implementation 08/2009
- minimale Implementation fertig 31.12.2009 :)
- brauchbare Implementation fertig 04/2010

IPv6 Allgemeinwissen



- IPv4 Adressen werden knapp
- größerer Adressraum, jedes System soll (ohne NAT) erreichbar sein. 32bit IPv4 vs. 128bit IPv6
- Syntax der Adressen ist anders
- Detailwissen:
  - aufgeräumt: div. Erweiterungen von IPv4 integriert
  - optimiert für Routing
  - feste Headerlänge

Teufel steckt im Detail

Syntax der Adressen

- Adressen sehen anders aus, gleiche Adresse wird auf verschiedene Arten ausgedrückt
  - ffff::7f00:1
  - ffff:0:0:0:0:0:7f00:1
  - ffff:0000:0000:0000:0000:0000:7f00:0001
  - ffff::127.0.0.1
- neue Syntax bei diversen Protokollen
  - http://[ip6addr]:port bei URLs
  - user@[IPv6:ip6addr] bei SMTP
  - in X509 Zertifikaten binär, muss man auf die Länge schauen
  - ...

Socket API

- Socket API für die meisten Anwendungen analog
- diverse spezielle Sachen (flow, scope, multicast..)
- RAW Socket API total anders
  - raw sockets empfangen keine Header
  - und können auch keine mittels IP_HDRINCL versenden
  - muss man alles mit [gs]etsockopt, sendmsg und recvmsg machen

Routing

- Ziel war Tabellen in den Routern klein zu halten
- Idee ist, das einem die Adressen nicht mehr gehören
- sondern man bei Providerwechsel einen neuen Prefix zugewiesen bekommt
- und eine Zeitlang alter und neuer Prefix parallel existieren
- welche Rechner, Applikationen oder Firewalls.. sind wohl auf sowas vorbereitet?

Autokonfiguration, DAD...

- jeder Rechner ermittelt mittels Duplicate Adress Detection (DAD) eine passende Adresse im Netzwerk
- DAD greift auch bei manueller Konfiguration
- früher war bei doppelter Adresse erratisches Verhalten, konnte schnell durch Umkonfiguration des duplicate Host behoben werden
- mit IPv6 geht erstmal garnichts mehr, selbst wenn der duplicate Host weg ist

Probleme in der Übergangsphase

- Applikationen brauchen Socket für IPv4 und IPv6
  - oder mapped IPv4
  - was in OpenBSD explizit nicht supported ist
- DNS: wenn Host AAAA und A Records hat, welches nimmt man?
  - auch wenn Host IPv6 Adressen hat muss IPv6 nicht funktionieren
  - evtl. nur im lokalen Netzwerk IPv6 aber nicht nach draussen
  - IPv6 kann wegen Tunnellösung langsamer sein bzw. erratisch funktionieren
  - evtl. gibt es Router auf dem Weg, die kein IPv6 verstehen
- Durchtunneln von Firewalls mittels IPv6 in IPv4 Tunnel, evtl. ohne das man sich dessen bewusst ist

IPv6 und Perl

- Situation nicht besser als bei Vortrag vom letzten Jahr :(
- kein IPv6 im CORE, auch in perl5.12 nicht
- Socket6 funktioniert
- IO::Socket::INET6 funktioniert überwiegend und wird eifrig weiter gepflegt
- Net::IPAddr zum Manipulieren/Parsen von Adressen funktioniert
- wichtig CORE und nicht CORE Module (Net::SMTP, Net::FTP, LWP...) können kein IPv6
- aber Workarounds existieren (Net::INET6Glue)

Migration des GeNUGate

- Administration
- DNS Preferenz
- Kernel
- ALGs
- diverse Programme
- Testumgebung

Administration über GUI

- keiner will mit langen IPv6-Adressen manuell arbeiten, d.h. hier wurden Aliase eingeführt, die statt der Adresse/Netzwerk verwendet werden können
- nach langer Diskussion entschieden keine explizite Unterstützung für Prefixwechsel zu haben, die Konkurrenz hat sowas auch nicht

DNS Preferenz - Optionen

- IPv6 vor IPv4
- IPv4 vor IPv6
- abhängig vom Prefix/Netzwerk machen
- jede Rule kann verschiedene Einstellungen haben

DNS Preferenz - Beschränkungen

- OpenBSD System resolver kann nur IPv4 vor IPv6 oder umgekehrt
- squid Resolver macht immer IPv6 vor IPv4
- ALG Resolver kann noch kein IPv6
- rtpproxy Resolver kann auch kein IPv6
- Nutzer muss Verhalten des Systems verstehen können

DNS Preferenz - Entscheidung

- lange Diskussionen
- wir gehen davon aus, das GeNUGate in stabilem Umfeld eingesetzt wird, d.h. keine instabilen IPv6 Tunnel
- daher: einheitlich IPv6 vor IPv4 wenn IPv6 konfiguriert
- allerdings kann bei den ALGs durch ACLs Netzwerke verbieten, dadurch kann man für bestimmte Netzwerke IPv4 erzwingen

Anpassungen OpenBSD Kernel

- IPv6 in OpenBSD stabil
- GeNUGate spezifische Erweiterungen mussten für IPv6 hinzugefügt werden
  - divert sockets (analog zu REDIRECT bei iptables) erweitern
  - Härtungen übernehmen (Spoofing checks, kein Forwarding)
- kleinere Bugs fixen

Anpassungen ALGs

- ACLs um IPv6 Support erweitern
- um Probleme mit DNS abzufangen werden jetzt wenn nötig alle Adressen durchprobiert, die DNS Lookup liefert
- Berücksichtigung von IPv6 in Protokollen (URLs, E-Mail Adressen...)
- Erweiterung FTP ALG für IPv6 (EPRT,EPSV)
- Anpassung Ping und Raw-IP ALG an neues RAW Socket API
  - Hoplimit etc via recvmsg lesen, statt IP-Header zu parsen
  - Hoplimit etc via sendmsg/setsockopt setzen, statt mit IP_HDRINCL
  - src + dst für icmp6 TIMEX per bind/connect setzen, statt mit IP_HDRINCL
- SMTP: erstmal keine Spamchecks bei IPv6, da nicht ganz klar ist, wie diese funktionieren sollen (z.B. RBL)

weitere Programme auf GeNUGate

- sendmail: funktioniert einfach
- squid kann eigentlich seit 3.x IPv6, aber
  - nur mit IPv4 compatible, was OpenBSD nicht unterstützt
  - Version 3.x ist sehr instabil (C++ rewrite)
  - daher erstmal kein IPv6 mit squid (geht auch ohne squid)
- OSPF
  - gated unterstützt kein IPv6 und wird es wohl auch nie mehr bekommen
  - evtl. benutzen wir OpenOSPFD
- rtpproxy
  - uralte Software, nur ein Kunde braucht das
  - eigener Resolver, den wir schon für IPv4 repariert hatten
  - Entscheidung: bekommt kein IPv6 Support

Testtools

- Net::SMTP, Net::FTP, LWP.. haben keine IPv6 Unterstützung, aber dank Net::INET6Glue kann man sie benutzen
- bisher dnet benutzt um für Tests kaputte IP Pakete zu kostruieren
  - aber das unterstützt kein IPv6
  - scapy kann IPv6 aber kein aktueller OpenBSD Port (viel Abhängigkeiten)
  - Net::Packet vom Autor als obsolet markiert
  - evtl. Nachfolger Net::Frame benutzen

weitere Erfahrungen

- unter OpenBSD macht ein getaddrinfo(ipv4-addr,AF_INET6) ein AAAA Lookup nach der ipv4-addr (RFC nicht klar in dieser Hinsicht)
  - sowas wurde von IO::Socket::INET6 <2.58 getriggert
  - gefixt durch Optimierung in IO::Socket::INET6
- 'localhost' nicht überall das gleiche
  - i.A auf ::1 und 127.0.0.1 auf IPv6+IPv4 fähigen Systemen
  - aber bei Fedora 10 ist ::1 localhost6
  - brach multihomed Test für IO::Socket::INET6

Fragen?