D N S s e c

HOWTO

Secure the Domain Name System

DENIC Technisches Meeting
28./29. September 2004

Holger.Zuleger@hznet.de

Agenda

- Einführung
  - Secure DNS Anwendungen
  - Historie
  - Resource Records
- Authentisierte Zonen Transfers
- Signierte Zonen
  - Schlüsselerzeugung
  - Signieren
  - Secure Resolver
- Secure Entry Points: Insellösung oder Hierarchien
- DNSsec und Privacy: NSEC und die Folgen

Was ist DNSsec?

- Secure DNS addressiert unterschiedliche Teilbereiche:
  a. Authentisierte Zonentransfers
  b. Sichere dynamische Updates (RFC3007)
     Interessantes Thema... aber nicht jetzt!
  c. Signierte Zonen
     Kryptographische Signatur über die Zoneninhalte
  d. Authentisierte Querys
     TSIG zwischen Stub-Resolver und Caching DNS
- Implementierung: bind-9.3.x, NSD 2.1.x
- Achtung: Unterschiedliche Verfahren
  — RFC2535 erläutert das Prinzip
  — Ergänzt durch RFC3658 (DS) und diverse Drafts
- Bedeutung nimmt zu / Mehr Dienste im DNS
  (Anti SPAM (MARID), ENUM, SSH-Fingerprints, SRV-Records)

DNSsec − History

Siehe: http://www.nlnetlabs.nl/dnssec/history.html

1983	„Erfindung“ des DNS durch Paul Mockapetris
1986	Formale Definition: RFC 1034 und 1035
1990	Steven Bellovin beschreibt Angriffsszenarien gegen DNS „Using the Domain Name System for System Break-ins“
1995	Veröffentlichung des Bellovin Paper und Beginn der Entwicklung an DNSSEC
1999	RFC 2535 wird veröffentlicht, findet allerdings keine Verbreitung
2001	Entwicklung von Verfahren für einfacheres Key Handling (DS)
2002	Neufassung des DNSsec Standard RFC2535bis draft-ietf-dnsext-dnssec-intro, draft-ietf-dnsext-dnssec-records, draft-ietf-dnsext-dnssec-protocol
2004	Veröffentlichung des überarbeiteten Standards bis Ende 2004 ?!

DNSsec − Resource Records

- TSIG (RFC2848)
  Transaktions Signaturen (Hashed MD5)
  Authentisierte Zonentransfers sowie sig. Querys und Updates

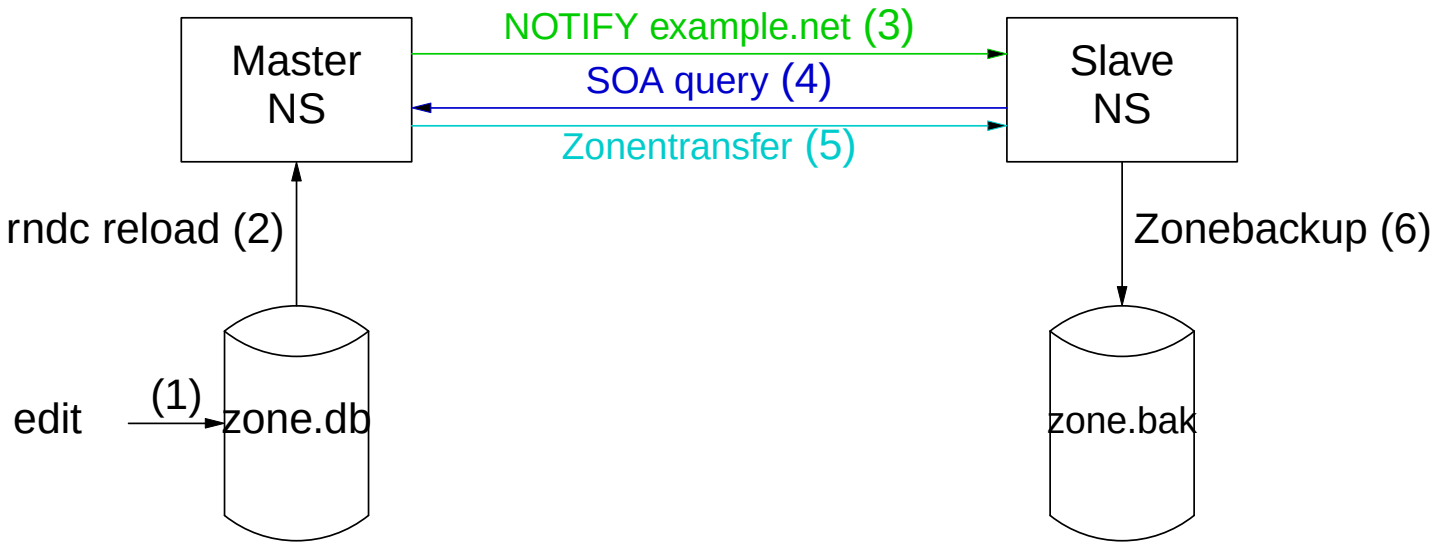
- TKEY (RFC2930)
  Verfahren zur Aushandlung von symmetrischen Keys für TSIG
  (Diffie-Hellman, Sig(0), GSSAPI)

- SIG(0) (RFC2931)
  Transaktions Signaturen (Public-Key: RSA-MD5, RSA-SHA1, DSA)
  Authentisierte dynamische Updates und TKEY Request

- RRSIG, DNSKEY, NSEC (RFC2535bis, RFC3845)
  Ehemals: SIG, KEY, NXT
  Signierte Resource Records (Public Key Verfahren)

- DS (RFC3658)
  Delegation der Vertrauensbeziehung

Zonentransfer



Angriffsszenarien:

- Verkörperung des Master (IP-Spoofing)
- Einschleusen falscher Daten beim Zonentransfer (man in the middle)

Transaktions Signaturen (TSIG)

- Shared Secret (HMAC-MD5) zwischen zwei(!) Hosts.
- Zur Zeit primär zwei Anwendungen:
  - Authentisierter Zonentransfer zwischen Master und Slave.
  - Dynamische Updates (z.B. DHCP-Server zu Master NS)
- Keine Verschlüsselung!
- Öffentliche Daten!
- Authentisierung und Integritäts-Check.
  - Bist du der Master Server?
  - Sind die Daten beim Transport modifiziert worden?
- Voraussetzung: Synchronisierte Uhren (max. diff. 5 Min)!

TSIG Konfiguration

5 Schritte zu sicheren Zonentransfers:

- Uhren synchronisieren (ntp)
- Generieren eines Shared Keys
  - Als Algorithmus ist zur Zeit nur HMAC-MD5 möglich
  - Keylänge muß zwischen 1 und 512 liegen
  - Der Name wird per Konvention aus den beiden FQDNs gebildet

```
$ dnssec-keygen -a HMAC-MD5 \
                -b 128 -n HOST \
                ns1.example.net-ns2.example.net
Kns1.example.net-ns2.example.net.+157+19512
```

Output: Zwei Dateien (leider)!

```
Kns1.example.net-ns2.example.net.+157+19512.key
Kns1.example.net-ns2.example.net.+157+19512.private
```

TSIG Konfiguration (2)

- Definiton des Key in `named.conf` (Master und Slave)

```
key "ns1.example.net-ns2.example.net." {
     algorithm hmac-md5;
     secret "/EB/vJJkokIoSGD6Wjmyeg==";
};
```

Das Secret holt man aus der `*.private` Datei

```
$ grep Key: Kns1.example.net-ns2.example.net.+157+19512.private
Key: /EB/vJJkokIoSGD6Wjmyeg==
```

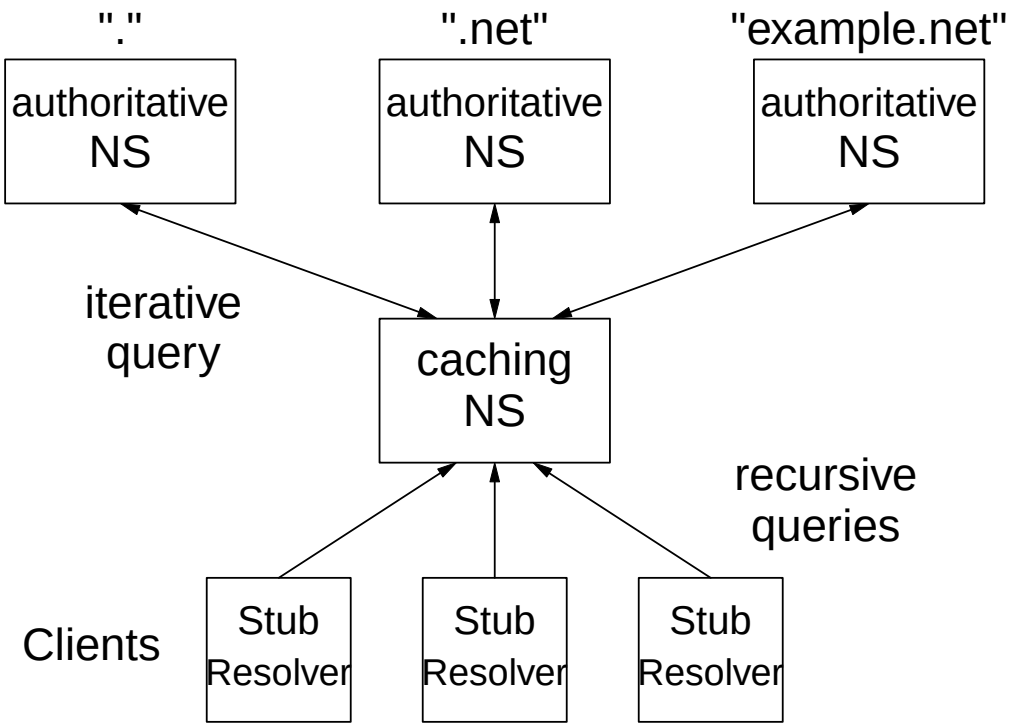
- Binding: Host to Key (mind. beim Slave)

```
server ip.addr.des.host {
     keys { key ns1.example.net-ns2.example.net.; };
};
```

- Access Control (Optional):

```
zone "example.net" IN {
     ...
     allow-transfer { key ns1.example.net-ns2.example.net.; };  # Master
     allow-transfer { none; };                                  # Slave
};
```

Authentisierte Namensauflösung



- Host zu Host Authentisierung ist nicht praktikabel
- Stattdessen: Signatur des Zoneninhalts (authenticated data origin)

Signierte Zonen

Authoritative NS (Master)

- Generieren eines asymetrischen Keys (Zonenkey)
- Einfügen des (öffentlichen) Keys in die Zone
- Erhöhen der Seriennummer im SOA Record
- Signieren der Zone (Offline)
- Reload der Zone

Resolving NS

- DNSsec einschalten
- Öffentlicher Zonenkey als „Secure Entry Point“ (SEP) hinterlegen

DNSKEY − Schlüssel zu signierten Zonen

Kommando `dnssec-keygen` mit den folgenden Parametern:

- Schlüsselalgorithmus und Keylänge
  - DSA (Keysize 512 bis 1024 Bit)
  - RSAMD5 (Keysize 512 bis 4096 Bit)
  - RSASHA1 (Keysize 512 bis 4096 Bit)
- Namenstyp: ZONE
- Schlüsselname = Domainname

```
$ dnssec-keygen -a RSASHA1 -b 512 -n ZONE sec.example.net
Ksec.example.net.+005+62759
                   ^   ^

Algorithmus ------+   |
Key ID --------------+
```

Zwei Dateien:

```
-rw-r--r--  1 hoz hoz  125 Aug 07 12:31 Ksec.example.net.+005+62759.key
-rw-------  1 hoz hoz  549 Aug 07 12:31 Ksec.example.net.+005+62759.private
```

Einfügen des Keys in die Zone

- Der öffentliche Teil des Keys steht als RR in der Datei K*.key:

```
sec.example.net. IN DNSKEY 256 3 5 AQPUSMEKBKBsYO/xd...
                           ^   ^ ^ ^

 Flags: Bit8 == Zonenkey --+   | | |
 Protokoll (3 == DNS) ---------+ | |
 Algorithmus --------------------+ |
 Schlüsselmaterial (gekürzt) ------+
```

- Dateiname des Keys ändert sich bei Neugenerierung

```
$ cat Ksec.example.net.+00*.key > keys.db
```

- Einfügen der Keys in die Zone ($INCLUDE Anweisung)

```
$ cat zone.db
@   7200 IN  SOA  ns1.example.net. hostmaster.example.net. ....
         IN  NS          ns1.example.net.
         IN  NS          ns2.example.net.
$INCLUDE keys.db
    ....
```

- Erhöhen der Seriennummer !

RRSIG − Unterschriebene Resource Records

- Signieren der Zone durch `dnssec-signzone`

```
$ dnssec-signzone -o sec.example.net zone.db
zone.db.signed
```

  - Sortieren der RR-Sets
  - Einfügen der NSEC Records
  - Signieren jedes RR-Sets und Einfügen des Signatur Records (RRSIG)

```
$ cat zone.db.signed
    ...
sec.example.net.  7200 IN NS     ns1.example.net.
                  7200 IN NS     ns2.example.net.
                  7200 IN RRSIG  NS 1 2 7200 (
Sig. Lifetime                    20040906100802 20040807100802
Keytag+Name                      62759 sec.example.net.
Signaturdaten                    AK9adL3Ov7VkVLYoan/5CHUO..== )
```

- Vor Ablauf der Signatur erneut signieren! (Intervall: (End-Start)/4)

Reload der Zone

- Name des Zonenfiles in der `named.conf` eintragen:

```
zone "sec.example.net" in {
        type master;
        file "sec.example.net/zone.db.signed";
        allow-transfer { key ns1.example.net-ns2.example.net.;};
        notify yes;
};
```

- Zone neu laden

```
$ rndc reload sec.example.net
```

- Meldungen kontrollieren

```
$ tail -f /var/log/named
07-Aug-2004 13:38:43.198 general: info: zone sec.example.net/IN: \
                                        loaded serial 12 (signed)
```

Secure Resolver (Caching NS)

- BIND 9.3.x benutzen
- Secure DNS in `named.conf` einschalten

```
options {
        recursion yes;
        dnssec-enable yes;
};
```

- Secure Entry Point in der „trusted-keys“-Section hinterlegen

```
trusted-keys {
    "sec.example.net." 256 3 5 "AQPUSMEKBKBsYO/xdnL/j..."
};
```

- Wie ?

```
$ { echo "trusted-keys {";
    cat Ksec.example.net.+005+*.key  |
    sed -n 's/^\([a-zA-Z.]*\) IN DNSKEY \([0-9]*\) \([0-9]\) \
            \([0-9]\) \(.*\)/"\1\" \2 \3 \4 "\5"/p'
    echo "};"
} >> named.conf
```

Oder besser: http://www.hznet.de/dns/create-trusted-key-section

What next?

- „Secure Entry Points“ müssen auf sicherem Wege übermittelt werden!

  Wie? Skalierbarkeit?

- Jeder (secure) Resolver benötigt den öffentlichen Zonenkey von **allen** signierten Zonen!

  Lösung: Delegation Signer d.h. Aufbau einer Hierarchie ?
  oder: Delegated Verification ? (Sichere Inseln)

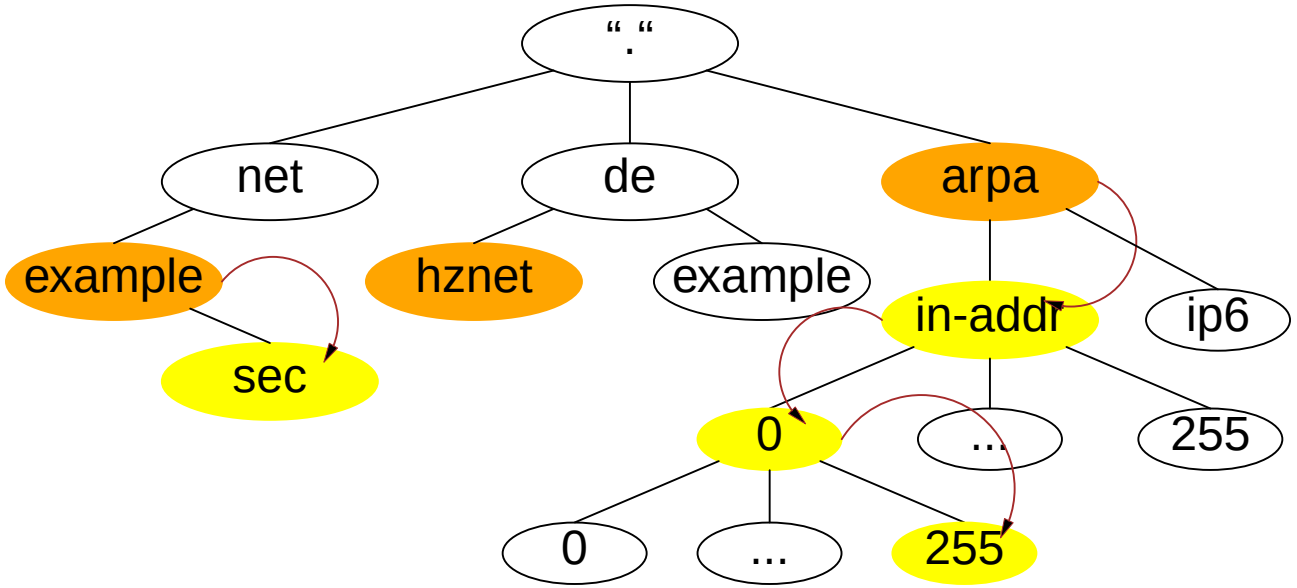
- Ändert sich der Zone Signing Key müssen alle Resolver angepasst werden. Lösung: Key Signing Keys (KSK)

- Wie kann eine negative Antwort signiert werden ?

  Lösung: NSEC Resource Records ?
  oder: Online signing ?
  oder: ????

- Signatur von Wildcard Records

Chain of Trust / Secure Entry Points



- Chain of Trust durch DS Records
- Wenige(r) „Secure Entry Points“
- Minimaler Schlüsselaustausch durch Key Signing Keys

KSK + ZSK

- Key Signing Keys (SEP)
  - Wird lediglich zum Signieren der Zonenkeys verwendet
  - Wenig genutzt (kleine Menge zu signierender Daten)
  - Große Schlüssellänge (DSA 1024, RSA 2048)
  - Lange Lebensdauer, d.h. selten geändert (1 Jahr?)
  - Änderung des KSK muß kommuniziert werden!
  - KSK über ein Bit im Flagfeld gekennzeichnet (RFC3757)
- Zone Signing Keys
  - Wird zum Signieren der Zonendaten verwendet
  - Häufig genutzt (große Menge zu signierender Daten)
  - Kleine Schlüssellänge (RSA 512 Bit)
  - Kurze Lebensdauer (Wenige Tage)
  - Änderung des Schlüssels muß nicht kommuniziert werden

KSK + ZSK (Konfiguration)

- Generieren des KSK (Option -f KSK)

```
$ dnssec-keygen -f KSK -n ZONE -a DSA -b 1024 sec.example.net
Ksec.example.net.+003+16004
```

- Generieren eines zweiten ZSK

```
$ dnssec-keygen -n ZONE -a RSASHA1 -b 512 sec.example.net
Ksec.example.net.+005+57764
```

- Einfügen der Keys in die Zone

```
$ cat Ksec.example.net.00[135]+*.key > keys.db
```

- Erhöhen der Seriennummer!
- Signieren + Neuladen

```
$ dnssec-signzone -o sec.example.net zone.db
zone.db.signed
$ rndc reload sec.example.net
```

- KSK in der trusted-key Section des Resolver eintragen!

DS – Delegation Signer

- Delegation: Einfügen eines Verweises auf den KSK in der Parentzone
  `dnssec-signzone` kreiert dsset- und keyset-Datei

- Die `keyset-`Datei enthält die DNSKEY-RR der Key Signing Keys
  Diese werden in der secure Zone (!) eingefügt

```
$ cat keyset-sec.example.net.
sec.example.net. 7200  IN DNSKEY 257 3 3 (
                       62uVBWg9spPDjXVaaXNaEwjLlNaKEqfwz4+A...
                       ) ; key id = 16004
```

- Die `dsset-`Datei enthält die DS-RRs als Verweis auf die KSKs
  Diese werden in der Parent-Zone (!) eingefügt.

```
$ cat dsset-sec.example.net.
sec.example.net.  IN DS 16004 3 1 55FBEE63...
 Key Tag ---------------^     ^ ^ ^
 Algorithm Number ------------+ | |
 Digest Type (SHA1) ------------+ +-- Hash des DNSKEY
```

- Beide Dateien müssen zum Parent übertragen werden

DS – Secure the Parent

- Der Parent muß seine Zone signieren!
- Wir brauchen Schlüsselmaterial für den Parent (KSK, ZSK, usw.)
- Signieren der Parent Zone

```
$ dnssec-signzone -g -o example.net zone.db
zone.db.signed
```

- Das Ergebnis:

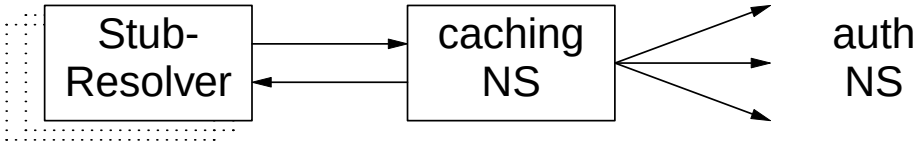
```
$ORIGIN example.net.
sec      7200   IN NS    ns1.example.net.
         7200   IN NS    ns2.example.net.
         7200   iN DS    16004 3 1 (55FBEE63... )
         7200   iN RRSIG DS 1 3 7200 20040906133208 (
                         20040807133208 65516 example.net.
                         dCzVu1NC7s/EB8e7Ynsl.... )
```

- Der Parent signiert nicht die Delegation (NS-Records)

  Lediglich der DS Record wird durch den Parent signiert!

- Resolver benötigt den SEP des Parent in der trusted-key Section

Stub-Resolver / Clients



Zwei Modi:

a. Signaturprüfung durch den Caching NS (Resolver)
   - EDNS0: do-Flag in der Anfrage setzen
   - EDNS0: UDP-Size 4096
   - In der Antwort sollte AD-Bit gesetzt sein
     (verified secure/insecure)

b. Eigenprüfung der Signatur
   - Stub-Resolver benötigt Trust-Anchor (SEP)
   - Zusätzlich bei der Anfrage das CD-Flag setzen
   - Die Antwort enthält auch Authority Section
   - AD-Bit nicht gesetzt

Stub-Resolver (dig)

```
$ dig @secResolver +multil +dnssec a.sec.example.net
; <<>> DiG 9.3.0rc3 <<>> @secResolver +multil +dnssec a.sec.example.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42021
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 4, AUTHORITY: 5, ADDITIONAL: 11

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;a.sec.example.net.                IN A

;; ANSWER SECTION:
a.sec.example.net.          5147 IN A 1.2.3.4
a.sec.example.net.          5147 IN RRSIG A 1 4 7200 20040906133347 (
                                 20040807133347 10809 sec.example.net.
                                 EZ0P5FVLaaRYx09Gh5VWVJzySt9CTPDhRgwAE522+L93
                                 27XecpZQwsKileKFdoExPQqQAWJJo4c9vUIZ+3tSBw== )
a.sec.example.net.          5147 IN RRSIG A 1 4 7200 20040906133347 (
                                 20040807133347 32998 sec.example.net.
                                 Ju5aWfSFGHppl+spF4/PVmB6vOZ/LBJQJqjGF/Du/tyS
                                 gNvUdsGkNn0EN2hxp8Z6FByTOKrV1w4SQZufBs0EVw== )
;; Query time: 107 msec
;; SERVER: 1.2.3.105#53(secResolver)
;; WHEN: Sat Aug 07 15:37:51 2004
;; MSG SIZE  rcvd: 2458
```

NSEC ...

Wie kann eine negative Antwort (offline) signiert werden?

- NSEC (früher NXT) Pseudorecord (Next SECure Record)
  - Alle Records sortieren
  - Bei jedem Label ein NSEC als Zeiger auf das nächste Label einfügen
  - Signieren der Zone

```
example.net.      SOA  ns1.example.net. ...
                  NS   ns1.example.net.
                  NS   ns2.example.net.
                  NSEC a.example.net. NS SOA RRSIG NSEC DNSKEY

a.example.net.    A    1.2.3.4
                  NSEC  b.example.net. A RRSIG NSEC

b.example.net.    A    1.2.3.5
                  NSEC example.net. A RRSIG NSEC
```

- Funktioniert auch mit Wildcards

... und die Folgen

- Ermöglicht einfaches Auslesen aller Labels einer Zone (Zonewalk)

  Prinzipielle Arbeitsweise:

```
$ dig +noall +answer nsec example.net
example.net.    7171  IN NSEC    a.example.net. A RRSIG NSEC

$ dig +noall +answer nsec a.example.net
a.example.net.  7171  IN NSEC    b.example.net. A RRSIG NSEC

$ dig +noall +answer nsec b.example.net
b.example.net.  7171  IN NSEC    example.net. A RRSIG NSEC
```

  Siehe auch: DNSSEC Walker (http://josefsson.org/walker/)

- Alternativen zu NSEC werden zur Zeit noch diskutiert

  draft-ietf-dnsext-dnssec-trans-00.txt

Zusammenfassung

- Authentisierter Zonentransfer
  - Einfach zu implementieren
  - Zeitsynchronisation!
- Signierte Zonen (authenticated data origin)
  - Grundlegende Werkzeuge vorhanden
  - Integration in vorhandene Provisionierungsabläufe notwendig
  - Keymanagement fehlt (Austausch Zonenkey)
  - Auch bei großen Zonen möglich (Offline, Multipozessor fähig)
- Secure Resolver
  - SEP Schlüsselverteilung und Austausch problematisch
  - Einsatz in definierten Umgebungen heute bereits machbar
  - Ausblick: Lookaside Domain

References

Miek Gieben, Internet Protocol Journal (Vol. 7, Issue 2, June 2004)
„DNSSEC: The Protocol, Deployment and a Bit of Development“

Nominum
BIND v9 Administrator Reference Manual

Olaf Kolkman, Ripe-NCC DISI
„DNSSEC Howto Version 1.3“

RFCs 1034, 1035, 2535, 2848, 2930, 2931, 3007, 3655, 3658, 3757, 3833, 3845

Drafts DNSSEC Operational Practices
draft-ietf-dnsop-dnssec-operational-practices-01.txt

DNSSEC Spezification
draft-ietf-dnsext-dnssec-intro-12.txt
draft-ietf-dnsext-dnssec-protocol-08.txt
draft-ietf-dnsext-dnssec-records-10.txt

Links http://www.dnssec.net
http://www.ietf.org/html.charters/dnsext-charter.html

Fragen ?

Fragen ?

http://www.hznet.de/dns/dnssec-denic040929.pdf

Fragen ?

http://www.hznet.de/dns/dnssec-denic040929.pdf

Herzlichen Dank für Ihre Aufmerksamkeit!

CONTENTS