D N S s e c

Sichere Namensauflösung im Internet

IT-Fachtage Nordhessen
24./25. September 2004

Holger.Zuleger@hznet.de

Agenda

- Einführung in das Domain Name System (DNS)
  - Anwendungsgebiete
  - Arbeitsweise
  - Bekannte Sicherheitsprobleme
- Secure DNS
  - Authentisierte Zonentransfers
  - Signierte Zonen
  - Secure Resolver
- More Secure Zones
  - Chain of Trust
  - Secure Delegation
- DNSsec und Privacy: NSEC und die Folgen

Was ist DNS?

- „Telefonbuch“ des Internet
  Welche IP-Adresse hat der Rechner `host.example.net`?
- Ersatz für die Datei `hosts.txt` aus der Anfangszeit des Internet
- Hierarchisch strukturiert, basiert auf Domainnamen:

```
host.subdomain.domain.tld
horst.mobile.example.net
```

- Primäre Anwendung:
  - Name zu IP-Adressauflösung (Forward Lookup)
  - IP zu Namensauflösung (Reverse Lookup)
  - Mailrouting (MX)
- In (naher) Zukunft
  - Mail Sender Authentisierung (MARID)
  - ENUM (Telefonnummer zu IP-Mapping)

Anwendungsgebiete

Die Informationen werden in der Domain als Resource Records gespeichert

- Namensauflösung IPv4 und IPv6

```
www.example.net.   A      1.2.3.4
                   AAAA   2001:0DB8:5678:1:210:5aff:feac:8b1b
```

- Reverse Lookups

```
4.3.2.1.in-addr.arpa.         PTR  www.example.net.
b.1.b.8.c....0.2.ip6.arpa.    PTR  www.example.net.
```

- Mailrouting (Mail to: `horst@example.net`)

```
example.net.   MX 10   mail.example.net.
               MX 20   smtp.example.com.
```

- Service Location Records (RFC2052)

```
_ldap._tcp.example.net.   SRV   10 1   389 ldap.example.net.
_sip._udp.example.net.    SRV    0 0   5060 sip.example.com.
```

- Public-Keys (KEY) und Zertifikate (CERT)

Anwendungsgebiete (2)

- E.164 Telefonnummer zu URI-Mapping, ENUM (RFC2916).
  Telefonnummer: +49 (0)6633/602022

```
2.2.0.2.0.6.3.3.6.6.9.4.e164.arpa.    \
    NAPTR 100 10 "u" "sip+E2U"  "!^.*!sip:info@ex.de!i"    .
    NAPTR 102 10 "u" "smtp+E2U" "!^.*!mailto:info@ex.de!i" .
```

- Anti SPAM Rules: z.B. Sender ID, SPF, usw.
  MTA Authorization Records in DNS (MARID)

```
example.net.   TXT  "v=spf1 mx mx:example.com ~all"
```

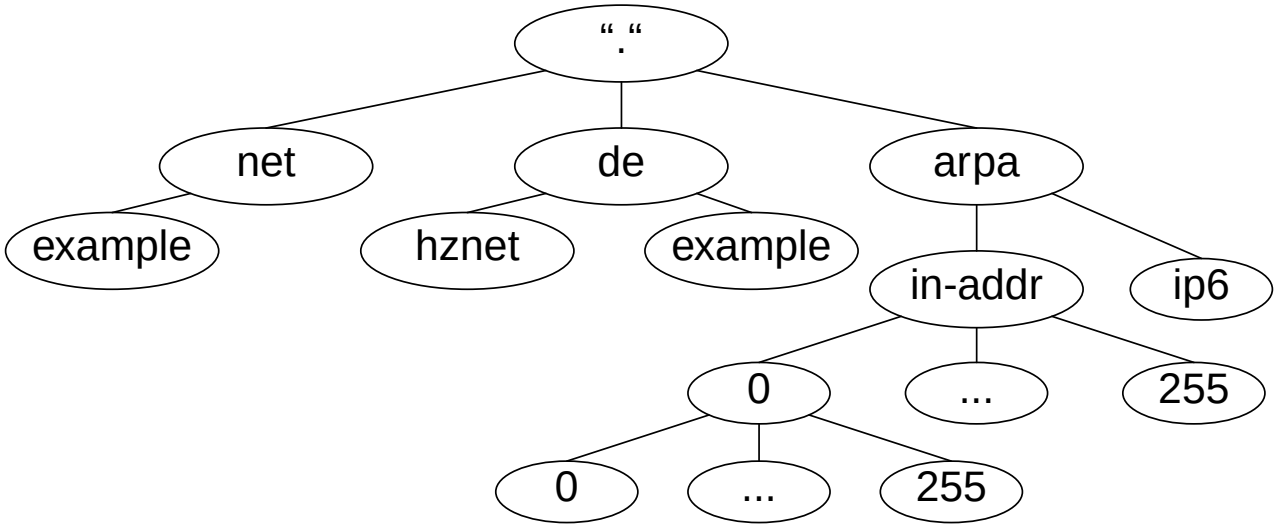
- SSH-Fingerprints (draft-ietf-secsh-dns-05.txt)

```
host.example.net.  SSHFP 2 1  123456789abcdef67890....
```

- Secure DNS (RFC2535) `DNSKEY`, `RRSIG`, `NSEC` und `DS`

  Später mehr davon

Arbeitsweise (Hierarchischer Aufbau)



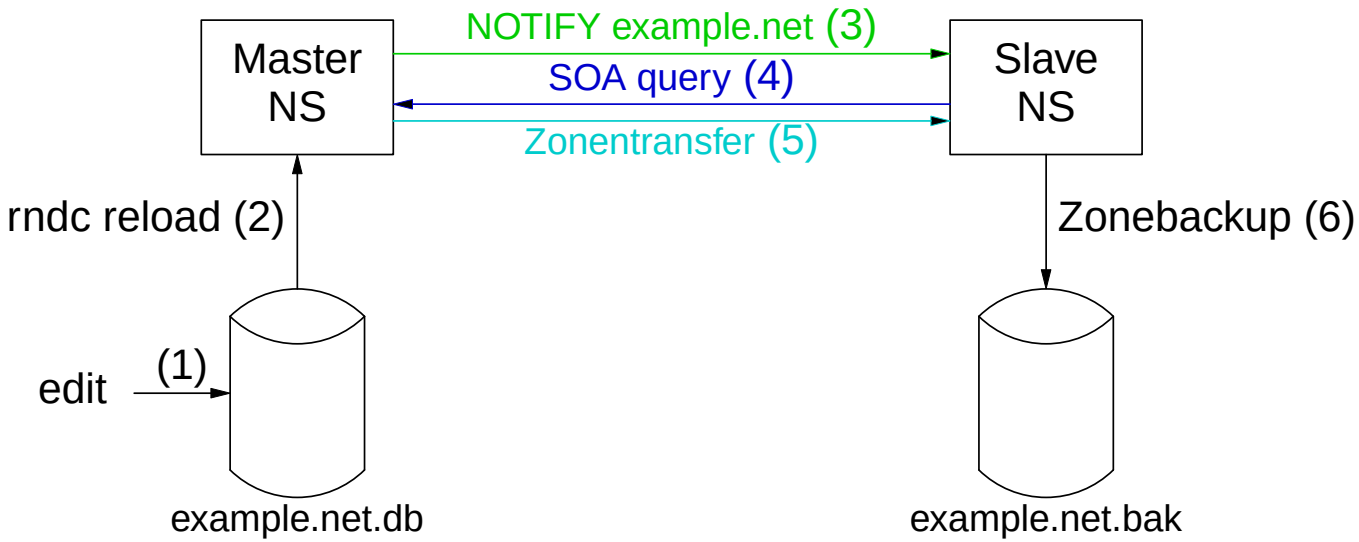
- Verkettung der Domains (Delegation) über NS-Records (top-down)

```
example.net.   NS   ns1.example.net.
               NS   ns1.example.com.
```

- Pro Domain mindestens zwei authorisierte Nameserver.
  Aber: Mehrere Domains können auf einem Nameserver gehostet werden.
- An der Spitze 13 Root-Nameserver
  `a.root-servers.net` … `m.root-servers.net`

Arbeitsweise (Zonentransfer)

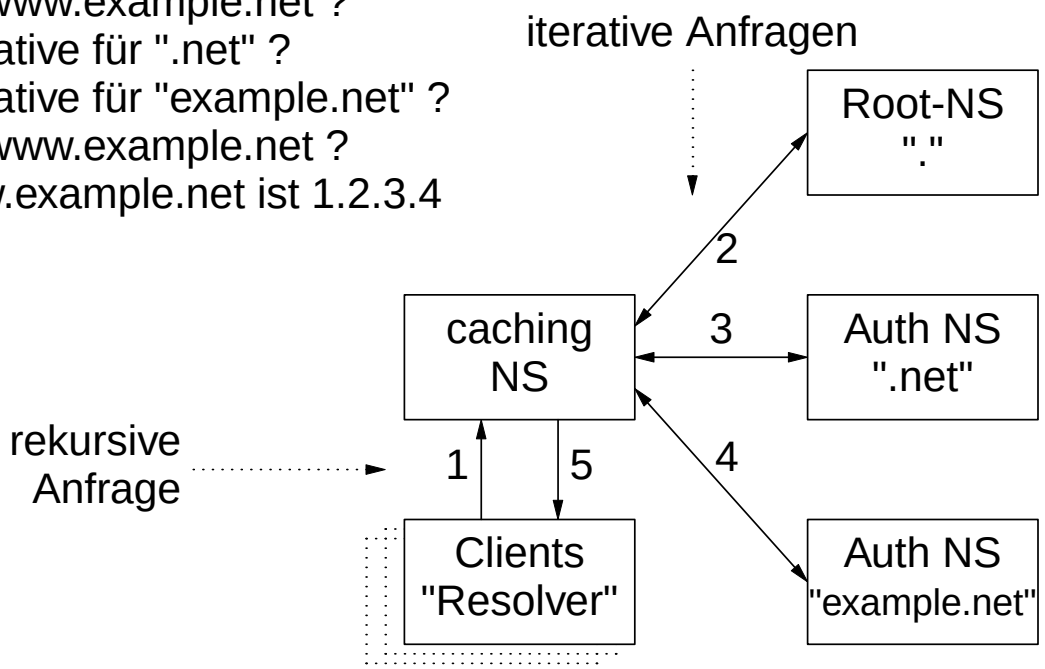
- Pro Domain mehrere authorisierte Nameserver
  Ein Master mehrere Slave Server
- Alle Änderungen ausschließlich auf dem Master
  Manuell oder durch „Dynamic Update“
- Übertragung der Änderungen auf die Slave Server (Zonentransfer)



Arbeitsweise (Namensauflösung)

- Clients (Stub-Resolver) verwenden Caching-NS (Resolver) zur Namensauflösung
- Der Resolving-Nameserver muß die IP-Adressen der Root-Server kennen!

1) Welche IP hat www.example.net ?
2) Wer ist authoritative für ".net" ?
3) Wer ist authoritative für "example.net" ?
4) Welche IP hat www.example.net ?
5) Die IP von www.example.net ist 1.2.3.4



Namensauflösung (Beispiel)

- Namensauflösung über lokalen (Caching) Nameserver

```
$ cat /etc/resolv.conf
nameserver 127.0.0.1
```

- Aufruf des `ssh`-Client (Stub-Resolver)

```
$ ssh horst.example.net
```

- Logfile (Querylog) des lokalen Nameservers

```
15:55:44.022 client 127.0.0.1#1027: query: horst.example.net IN AAAA +
15:55:46.474 client 127.0.0.1#1027: query: horst.example.net IN A +
15:55:46.721 client 127.0.0.1#1027: query: horst.example.net IN SSHFP +
```

- Logfile (Querylog) des remote Nameservers (sshd)

```
15:55:46.190 client 1.2.1.159#53: query: example.net IN DNSKEY -SE
15:55:46.354 client 1.2.1.159#53: query: horst.example.net IN AAAA -SE
15:55:46.487 client 1.2.1.159#53: query: horst.example.net IN A -SE
15:55:46.820 client 1.2.1.159#53: query: horst.example.net IN SSHFP -SE
15:55:47.538 client 127.0.0.1#3160: query: 159.1.2.1.in-addr.arpa IN PTR +
```

Namensauflösung (Tools)

- Einfach: `host`

```
$ host www.example.net
www.example.net has address 1.2.4.3
```

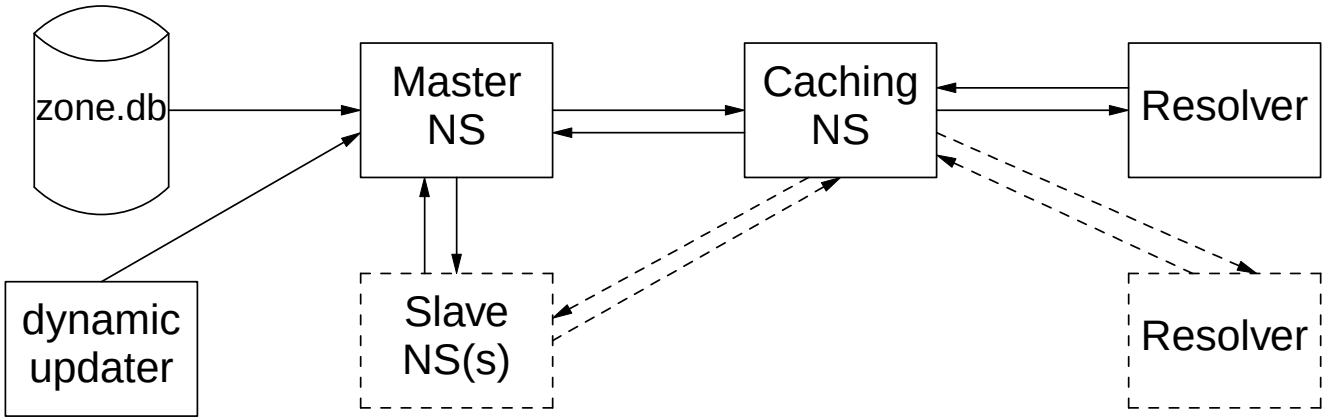
- Verbreitet, aber nicht für Fehlersuche geeignet: `nslookup`

```
$ nslookup www.example.net
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
Name:   www.example.net
Address: 1.2.4.3
```

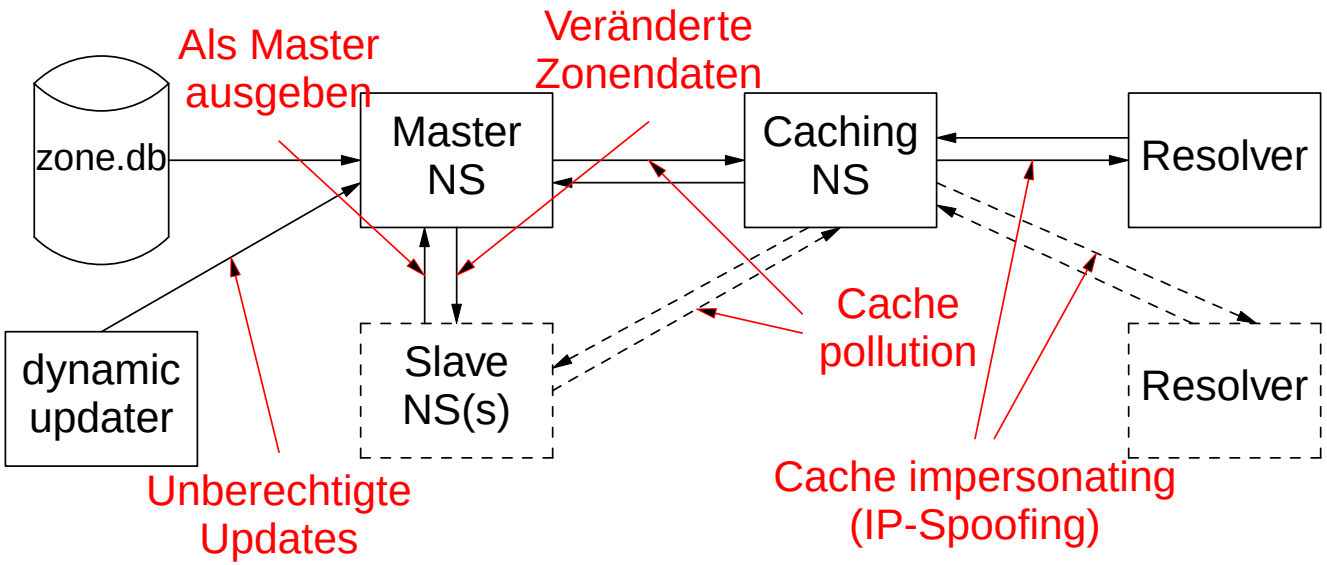
- Vielseitig und „kompliziert“: `dig`

```
$ dig www.example.net
; <<>> DiG 9.3.0rc4 <<>> www.example.net
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31926
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.example.net.                IN      A

;; ANSWER SECTION:
www.example.net.          1939    IN      A       1.2.4.3

;; AUTHORITY SECTION:
example.net.              81139   IN      NS      ns1.example.com.
example.net.              81139   IN      NS      ns1.example.net.

;; ADDITIONAL SECTION:
ns1.example.com.          167539  IN      A       1.25.240.148

;; Query time: 51 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; MSG SIZE  rcvd: 124
```

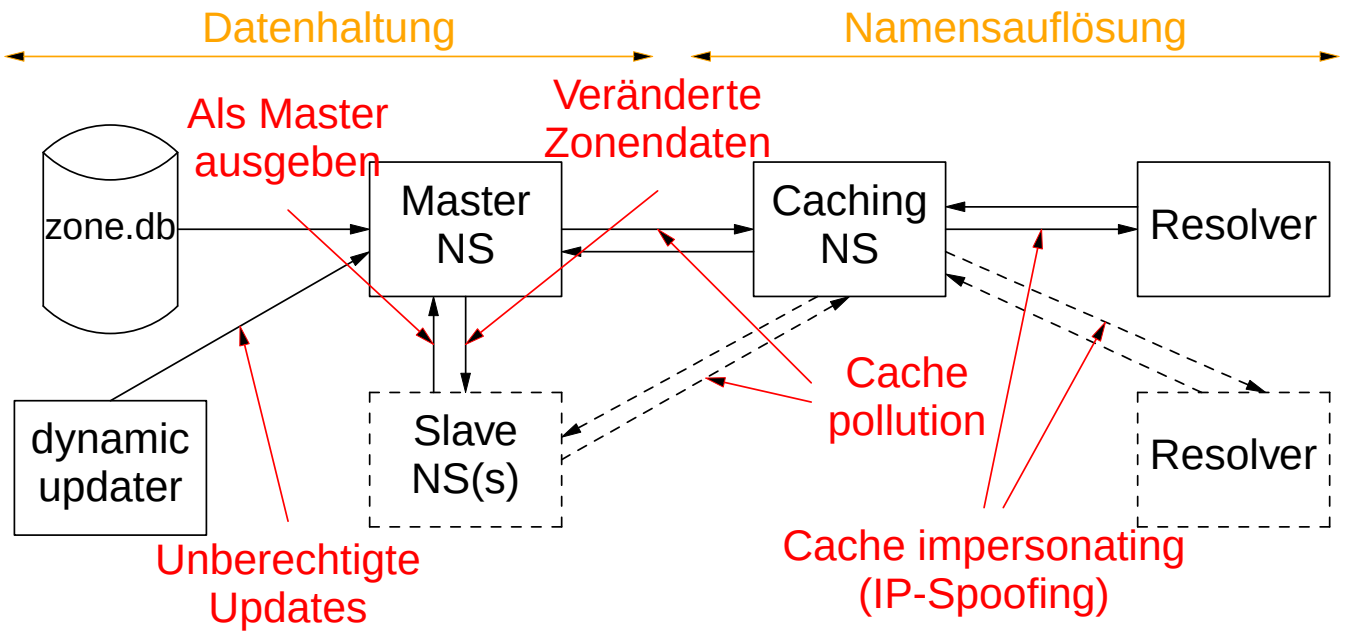
Angriffsszenarien



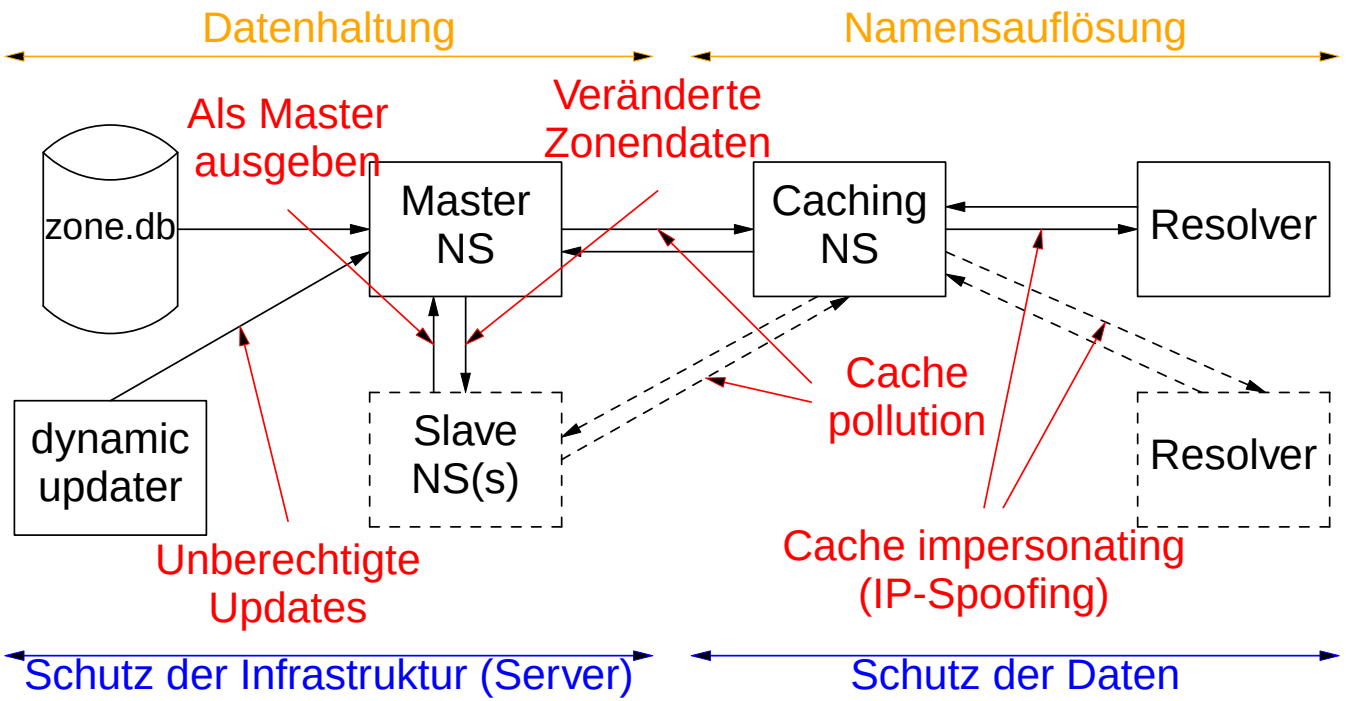
Angriffsszenarien



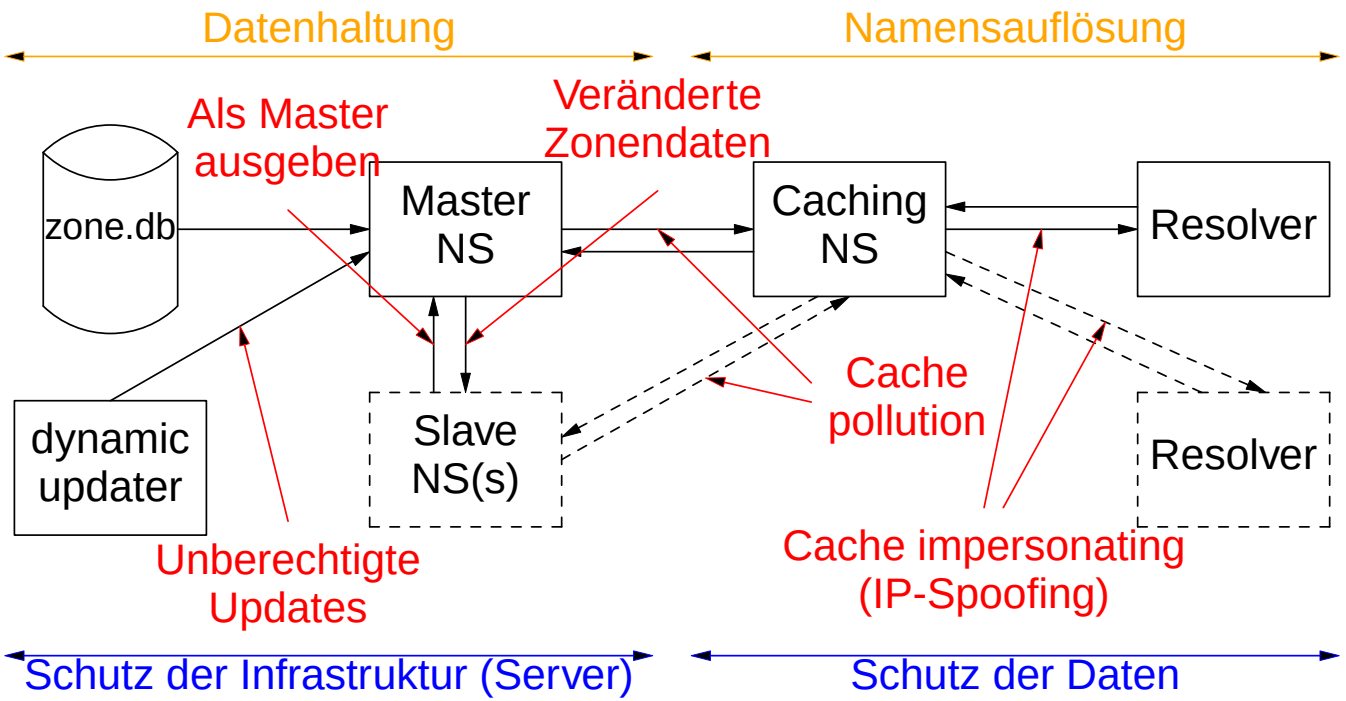
Angriffsszenarien



Angriffsszenarien



Angriffsszenarien



- Schutz der (Server) Infrastruktur
  Authentisierung über symmetrische- oder asymmetrische Keys (Public-Keys)

- Schutz der Daten (Resource Records)
  Signierte Datensätze (Public-Keys)

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

- Keine PKI
  In definierten Umgebungen als PKI einsetzbar (SSH-Fingerprints, IPsec-Keys)

Authentisierter Zonentransfer (TSIG)

- Shared Secret (HMAC-MD5) zwischen zwei(!) Hosts.
- Authentisierung und Integritäts-Check.
  - Bist du der Master Server?
  - Sind die Daten beim Transport modifiziert worden?
- Keine Verschlüsselung! Öffentliche Daten!
- Voraussetzung: Synchronisierte Uhren (max. diff. 5 Min)!
- Shared Secret erzeugen
  - Algorithmus: HMAC-MD5, Keylänge 1 bis 512 Bit
  - Key benötigt einen Namen: FQDN der beiden Hosts

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

Output: Zwei Dateien (.key und .private)
Beide (!) enthalten das Shared **Secret**

Transaktions Signaturen (Konfiguration)

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

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

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

```
$ grep Key: Kns1.example.net-ns1.example.com.+157+19512.private
Key: /EB/vJJkokIoSGD6Wjmyeg==
```

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

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

- Access Control (Optional):

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

Signierte Zonen

- Signatur basiert auf asymmetrischen Keys (Public-Key Kryptographie)
  z. Zeit DSA, RSA-MD5, RSA-SHA1; `dnssec-keygen` generiert die Schlüssel
- Mindestens zwei Keys pro Zone/Domain

  Zone-Signing-Key (ZSK): Signiert die Zonendaten
  - häufig genutzt (große zu signierende Datenmenge)
  - kleine Schlüssellänge
  - kurze Gültigkeit / häufiger Schlüsselwechsel

  Key-Signing-Key (KSK): Signiert nur die Zonenschlüssel
  - wenig genutzt (kleine zu signierende Datenmenge)
  - große Schlüssellänge
  - lange Gültigkeit / seltene Schlüsselwechsel ⇒ als SEP geeignet
- Signieren der Zone kann auf separatem Rechner durchgeführt werden.
  `dnssec-signzone` benötigt Zonenfile und die privaten Schlüssel
- Signaturen haben eine definierte Gültigkeit
  Vor Ablauf erneut signieren! Seriennummer ändern!

DNSKEY – Schlüssel zu signierten Zonen

Mit `dnssec-keygen` beide Schlüssel erzeugen:

- Als Schlüsselname wird der Domainname verwendet
- Wir verwenden DSA (1024Bit) für KSK, RSA-SHA1 (512Bit) für ZSK
- Identifizierung des Schlüssel durch Name, Algorithmus und Key ID

```
$ dnssec-keygen -f KSK -a DSA -b 1024 -n ZONE sec.example.net
Ksec.example.net.+003+16004
$ dnssec-keygen -a RSASHA1 -b 512 -n ZONE sec.example.net
Ksec.example.net.+005+62759
```

Pro Schlüssel zwei Dateien:

```
-rw-r--r--  1 hoz hoz  585 Aug 07 12:29 Ksec.example.net.+003+16004.key
-rw-------  1 hoz hoz  688 Aug 07 12:29 Ksec.example.net.+003+16004.private
-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
```

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

```
$ cat Ksec.example.net.+00[53]+*.key
sec.example.net. IN DNSKEY 257 3 3 CJYTDZ0l/aW5+...
sec.example.net. IN DNSKEY 256 3 5 AQPUSMEKBKBsYO/xd...
```

RRSIG – Unterschriebene Resource Records

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

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

```
$ cat sec.example.net.db
@   7200 IN  SOA  ns1.example.net. hostmaster.example.net. 1 ....
         IN  NS          ns1.example.net.
         IN  NS          ns2.example.net.
$INCLUDE sec.example.net.keys
```

- Erhöhen der Seriennummer

- Signieren der Zone durch `dnssec-signzone`

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

```
$ cat sec.example.net.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...== )
```

Reload der Zone

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

```
zone "sec.example.net" in {
        type master;
        file "sec.example.net.db.signed";
        allow-transfer { key ns1.example.net-ns1.example.com.;};
        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)
```

!Nicht vergessen: Resigning vor Ablauf der Signatur!

Secure Resolver (Caching NS)

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

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

- Wie kann der Resolver die Signaturen überprüfen?
  Public-Key (KSK) als „Secure Entry Point“ hinterlegen

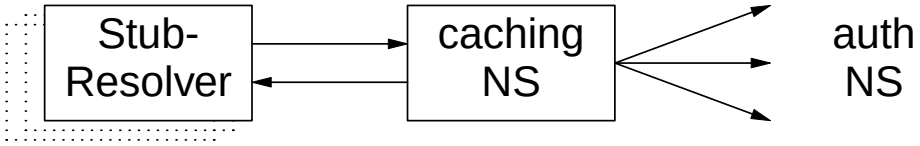
- `trusted-keys`-Section nimmt SEP(s) für diverse Domains auf:

```
trusted-keys {
    "sec.example.net." 257 3 3 "CJYTDZ0l/aW5+..."
};
```

- Eintrag aus der Public-Key Datei extrahieren:

```
$  cat Ksec.example.net.+003+*.key  |
   sed -n 's/^\([a-zA-Z.]*\) IN DNSKEY \([0-9]*\) \([0-9]\) \
           \([0-9]\) \(.*\)/"\1\" \2 \3 \4 "\5"/p'
```

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)

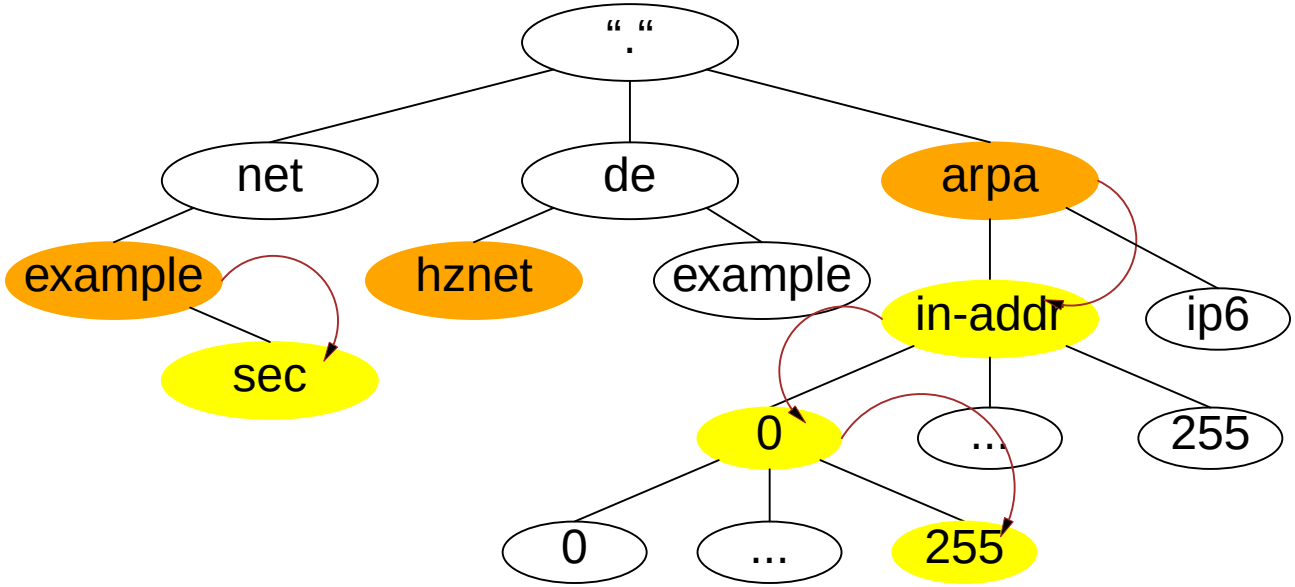
- Das Authenticated Data Flag (RFC3655) zeigt an, dass die Antwort kryptographisch geprüft wurde.

```
$ 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
```

More Secure Zones

- Secure DNS in definierter Umgebung
  - Kontrolle über Secure Resolver
  - Übermittlung der „Secure Entry Points“ auf sicherem Wege
  - Austausch des KSK erfordert Änderung des SEP
- Secure DNS bei unbekannten Zonen
  - Initiale Einrichtung des SEP?
  - Benachrichtigung über Änderung des KSK?
- Chain of Trust (hierarchische Vertrauensbeziehung)
  - Parentzone signiert den KSK (Delegation Signer Record)
  - Secure Resolver benötigt nur noch SEP des Parent
  - Weniger „Trusted Keys“ im Resolver
- Delegated Verification
  - SEP von Secure Zones werden in spez. Domain hinterlegt

Chain of Trust



- Chain of Trust durch DS Records
- Wenige(r) „Secure Entry Points“
- Delegated Verification bricht hierarchisches Trust Modell auf (Experimental)

DS – Delegation Signer

- Secure Delegation: Verweis auf die KSK der Secure Zone
- `dnssec-signzone` erzeugt `dsset-` und `keyset-`Datei
- In der secure Zone fügen wir die `keyset-`Datei ein (enthält den KSK)

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

- In der Parentzone wird der DS-Record eingetragen
  (DS-Record ist ein Verweis auf den DNSKEY in der Secure Zone)

```
$ 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 example.net.db
example.net.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

NSEC ...

Wie kann eine negative Antwort (offline) signiert werden?

- NSEC Pseudorecord (Next SECure Record)
  - Alle Records sortieren
  - Bei jedem Label einen NSEC-Record als Zeiger auf das nächste Label einfügen
  - Erstellen der Signatur Records

```
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
```

- NSEC-Records werden durch `dnssec-signzone` erzeugt

... 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, Key-Signing-Key)
  - Auch bei großen Zonen möglich (offline, multitasking fähig)
- Secure Resolver
  - SEP Schlüsselverteilung und Austausch problematisch
  - Einsatz in definierten Umgebungen heute bereits machbar
  - Chain of Trust: Wann?
  - 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, Oktober 2003
„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/dnssec/dnssec-itfn040925.pdf

Fragen ?

http://www.hznet.de/dnssec/dnssec-itfn040925.pdf

Herzlichen Dank für Ihre Aufmerksamkeit!

CONTENTS