DNS and DNSSEC

Luca Campa

University of Klagenfurt – Introduction to Cybersecurity

Why DNS Security Matters

- DNS is the first dependency for most Internet connections.
- If DNS answers are manipulated, users can be redirected before TLS starts.
- Failures affect both security and availability:
  - phishing, malware delivery, credential theft
  - denial of service at recursive resolvers
  - propagation through caching and delegation

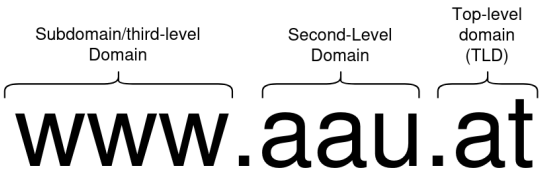
DNS Foundations

DNS (Domain Name System)

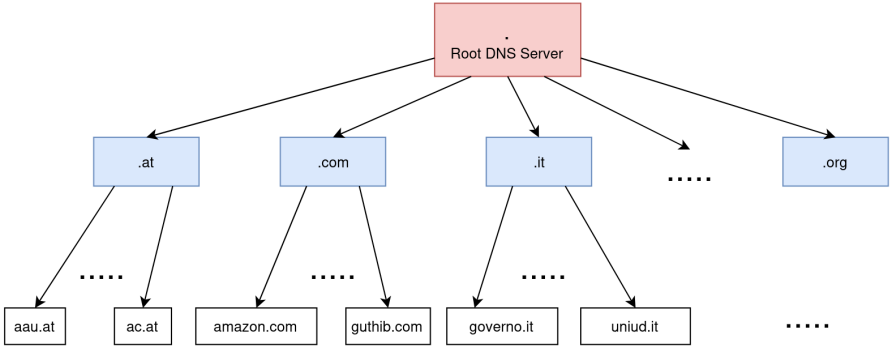
- DNS is the "address book" of the Internet, mapping human-friendly names to IP addresses and other resources.
- DNS maps domain names to resource records (A/AAAA, MX, NS, TXT, ...).
- Most queries are UDP/53 (fallback to TCP when needed).
- Resolution is hierarchical:
  - root → TLD → authoritative server
- Recursive resolvers cache responses to reduce latency and load.

Note: DNS **is eventually consistent, not instantly consistent**. TTL values control how long old answers may remain in caches.

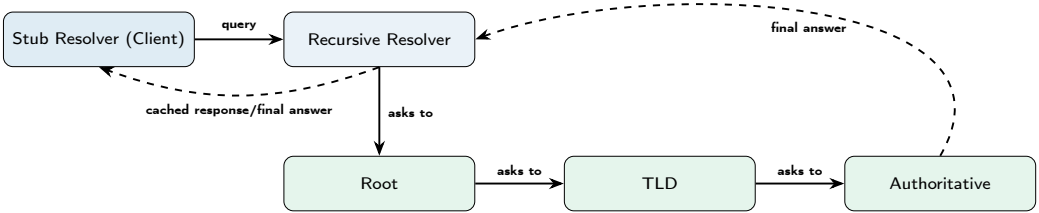
What are those domains?



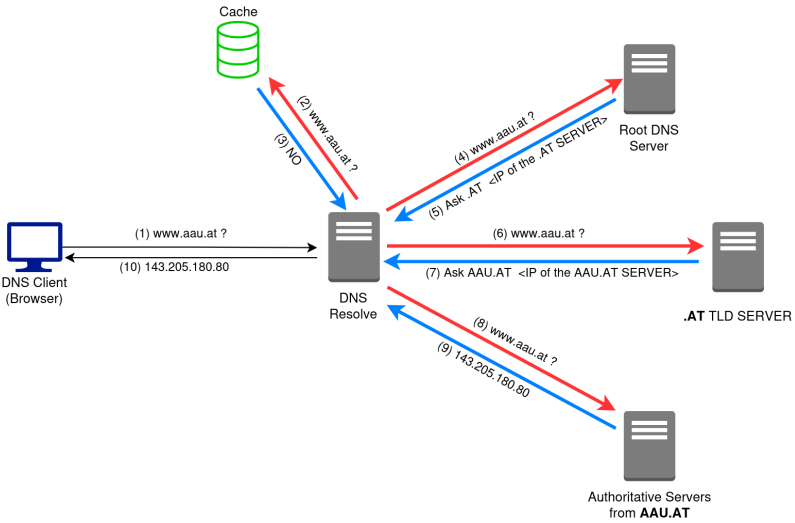
DNS Hierarchy



DNS Resolution Path

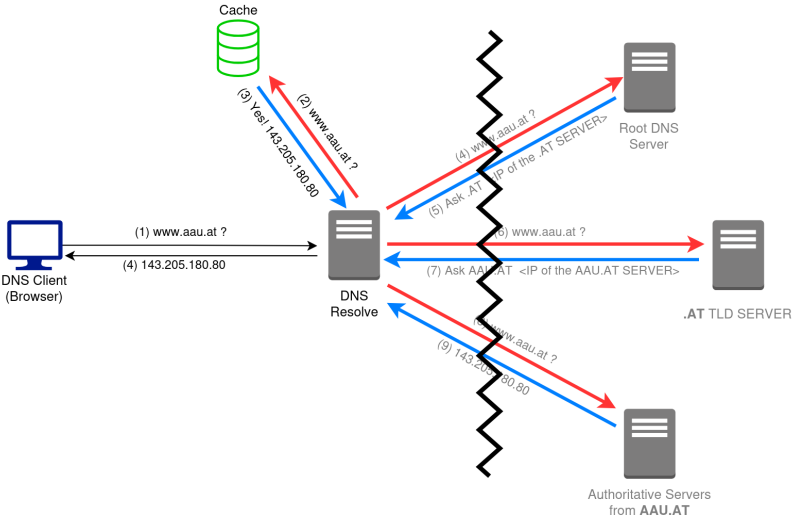


DNS Resolution Path (Not in cache)



Resolver stores answer in cache for TTL (time-to-live) duration.

DNS Resolution Path (in cache)



Weak Points

- Stub (the client) $\leftrightarrow$ resolver channel (local network risk).
- Resolver $\leftrightarrow$ authoritative path (on-path tampering risk).
- Resolver cache (poisoning and stale attack surface).
- Registrar and zone management (domain takeover risk).

Always ask yourself: *Which organization controls each boundary?*

Type of DNS Records

- **A/AAAA**: IPv4/IPv6 address mapping.
- **NS**: authoritative name servers for a zone.
- **MX**: mail exchange servers for a domain.
- **TXT**: arbitrary text data (often used for SPF, DKIM, DMARC).
- **CNAME**: alias to another domain name.
- **DNSSEC-specific**: DNSKEY, RRSIG, DS, NSEC/NSEC3.

Note: DNS is extensible, so new record types can be added without breaking existing functionality. This is both a strength and a potential attack vector if not properly managed.

CLI Commands for DNS Inspection

- `dig` (DNS lookup utility):
  - `dig example.com A` (basic query)
  - `dig +dnssec example.com A` (with DNSSEC records)
  - `dig +trace example.com A` (full resolution path)
- `nslookup` (interactive DNS client):
  - `nslookup example.com` (basic query)
  - `nslookup -type=DNSKEY example.com` (DNSSEC keys)
- `host` (simple DNS lookup):
  - `host example.com` (basic query)
  - `host -t DNSKEY example.com` (DNSSEC keys)

Example

dig aau.at

```
; «» DiG 9.20.11-1ubuntu2.1-Ubuntu «» aau.at
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41545
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;aau.at.                IN      A

;; ANSWER SECTION:
aau.at.         579     IN      A       143.205.180.80

;; Query time: 1 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Fri May 29 16:55:19 CEST 2026
;; MSG SIZE  rcvd: 51
```

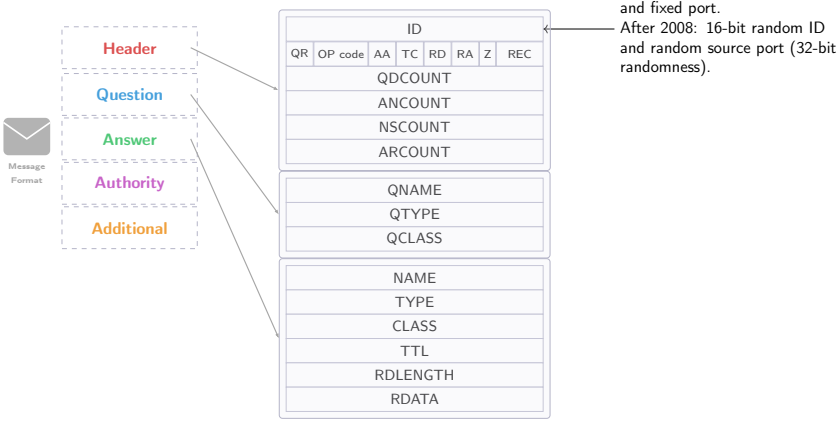
DNS Attacks and Vulnerabilities

Attack Taxonomy

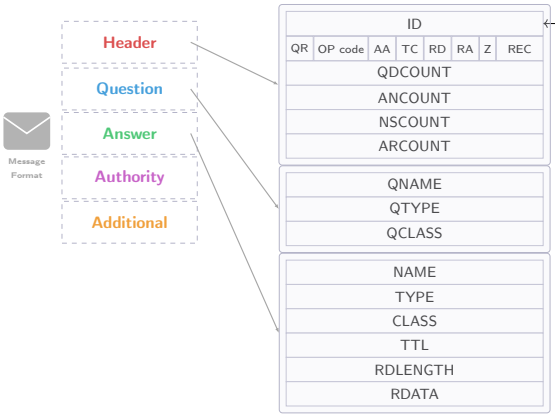
- **Integrity attacks**: inject forged responses, cache poisoning.
- **Availability attacks**: overload resolvers or authorities.
- **Confidentiality attacks**: observe or exfiltrate data via DNS.
- **Control-plane hijack**: change delegation, NS, or registrar settings.

Each class maps to a different defensive focus: validation, rate control, monitoring, and identity/access governance.

DNS Message Format



DNS Message Format



Before 2008: 16-bit random ID and fixed port.
After 2008: 16-bit random ID and random source port (32-bit randomness).

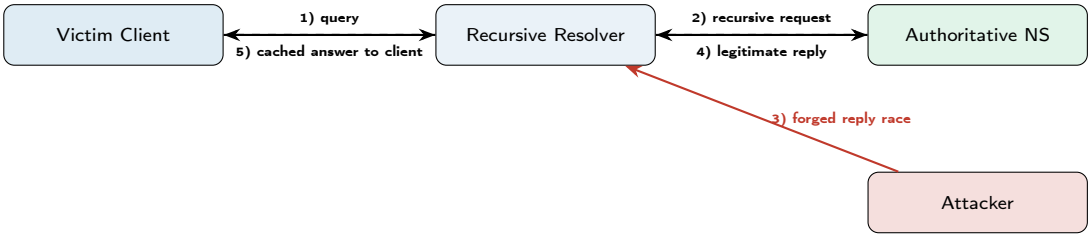
Question: where is the source port?

1) DNS Cache Poisoning (Classic)

- Goal: make resolver cache a forged record.
- Technique: race a spoofed response before the legitimate one.
- Historically exploited weak entropy in:
  - 16-bit transaction ID
  - predictable source ports
  - shared query behavior
- Impact: users redirected to attacker-controlled infrastructure.

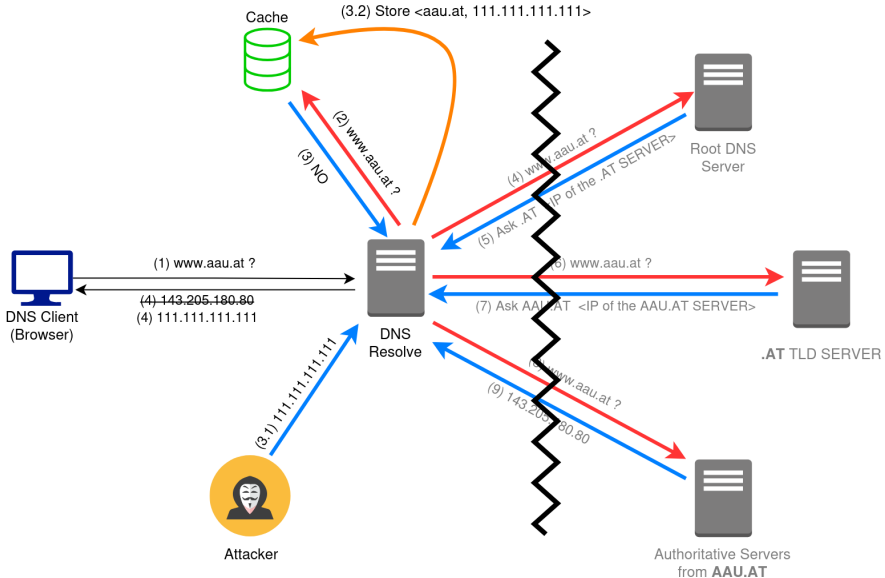
Note: one successful poison affects many users behind the same resolver.

1) DNS Cache Poisoning (Classic)



If step 3 arrives before step 4 and passes checks, the cache is poisoned.

1) DNS Cache Poisoning (Classic)

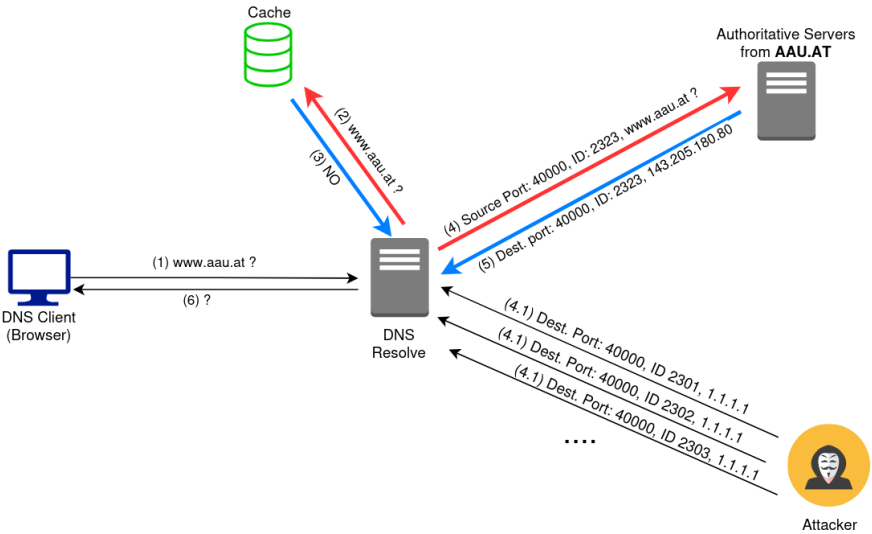


2) Kaminsky-Style Cache Poisoning (2008)

1. Trigger many cache misses for random subdomains.
2. Force resolver to ask authoritative servers repeatedly.
3. Flood forged referrals with guessed ID/source-port.
4. If one guess wins, malicious NS can be cached.

Defenses: source-port randomization, query hardening, DNSSEC validation.

2) Kaminsky-Style Cache Poisoning (2008)



Modern Variants: IP Fragmentation and SAD DNS

SAD DNS (2020):

- it showed side-channel methods can reduce spoofing uncertainty;
- Even with randomized ports, network ICMP/fragmentation behavior can leak information.
- After leaking the source port, normal Kaminsky-style poisoning can proceed with 16-bit ID guessing.

IP Fragmentation Attacks (2012):

- Trigger huge DNS responses which are then fragmented;
- Fragmentation is handled by the IP layer, hence the attacker can completely bypass UDP source port and DNS ID randomness, as it only requires guessing a 16-bit IP ID.

4) DNS Hijacking

- **Resolver hijack:** malware/router changes resolver settings.
- **Registrar hijack:** attacker changes NS records at registrar.
- **Authoritative compromise:** zone content directly modified.

- Changes can affect all users of the domain, not just one client.
- Weak points: missing MFA at registrar, weak API keys, and poor change-audit controls.

5) DNS Amplification and Reflection

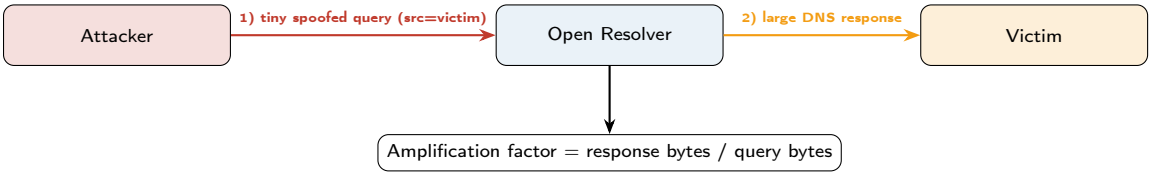
- Attacker spoofs victim IP in small DNS queries.
- Attacker asks the resolver for large responses (e.g., ANY queries, DNSSEC-signed records) but with the victim's source IP.
- Open resolvers send larger responses to victim.
- Amplification factor can be significant with DNSSEC/large responses.

Mitigations:

- eliminate open recursion
- response rate limiting (RRL)
- anti-spoofing at network edge.

If source spoofing is blocked, reflection attacks collapse.

5) DNS Amplification and Reflection



Mitigation: remove open recursion and block source-IP spoofing.

6) NXDOMAIN and Random-Subdomain Flooding

- Massive queries for Non-eXistent DOMAINS stress authority/resolver chains.
- Caching helps less because names are unique.
- DNSSEC negative proofs (we will see later) can increase packet and validation work (NSEC/NSEC3).

Mitigations: aggressive negative caching, rate limiting, delegation design hardening.

Common attack in DDoS campaigns against CDN domains and large SaaS platforms.

7) DNS Tunneling and Exfiltration

Attackers use it to bypass security systems and communicate with systems inside a private network.

- Data encoded into subdomains (e.g., base32/64 chunks) or in TXT records.
- Authoritative server controlled by attacker reassembles payload.

Detection signals:

- unusually long labels and high entropy strings
- abnormal TXT or NULL query patterns

Example: Suspicious Tunneling Pattern

```
dGhpc19pc190aGVfY291cnNl.attacker-domain.tld
b2ZfaW50cm9kdWN0aW9uX3Rv.attacker-domain.tld
Y3liZXJzZWN1cml0eV8yMDI2.attacker-domain.tld
```

Threat identifiers:

- long, high-entropy labels
- repeated queries to one rare authoritative domain
- TXT record lookups at regular intervals

DNSSEC Fundamentals

What DNSSEC Provides (and Does Not)

Provides:

- data origin authentication
- integrity of DNS RRsets
- authenticated denial of existence

Does not provide:

- confidentiality (no encryption)
- endpoint trust of web content itself
- protection from all DoS vectors

Think of DNSSEC as *signed DNS*, not encrypted DNS.

DNSSEC Resource Records

- **DNSKEY**: public keys for zone signing.
- **RRSIG**: digital signature over an RRset.
- **DS**: parent-to-child hash link for chain of trust.
- **NSEC/NSEC3**: authenticated denial of non-existence.

Validation fails if any required link (DS, DNSKEY, RRSIG, or time validity) is broken.

Public-Key Cryptography in DNSSEC

Each zone contains two public key pairs (private key and corresponding public key):

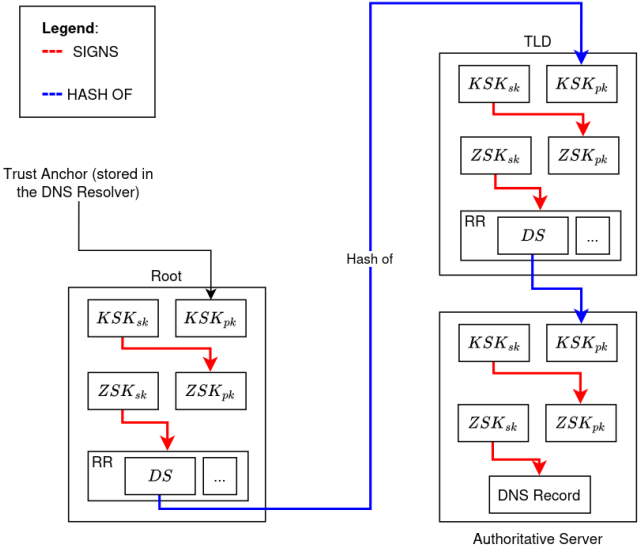
- **KSK (Key Signing Key)** signs DNSKEY RRset (the ZSK).
- **ZSK (Zone Signing Key)** signs zone data RRsets (rotated more often).
- Rollover must consider TTLs, RRSIG validity windows, Delegation Signer (DS) propagation.

Failure patterns:

- expired signatures
- missing DS during delegation changes
- inconsistent signer setups across providers

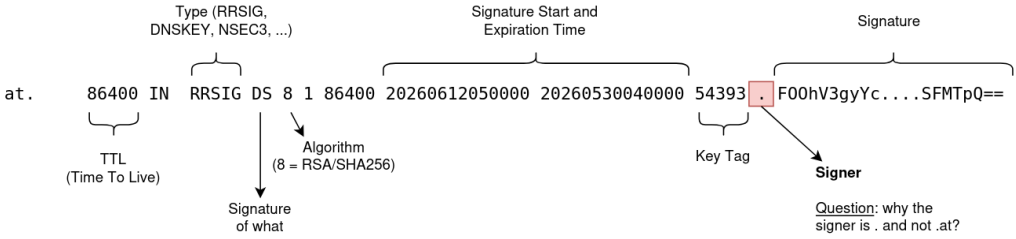
The DNS root zone authentication is carried out using a public key called the **Trust Anchor**, stored in the DNS resolver like a built-in certificate.

Chain of Trust (Generation)



- Every SIGN generates a RRSIG record
- RRSIG is stored within the zone
- Each zone is responsible of publishing its DS Hash to the upper layer

RRSIG Resource Record structure



DNSKEY record structure

Almost same as before, but with a flag field that indicates if the key is a KSK (257) or ZSK (256), and a protocol field that must be set to 3 for DNSSEC keys.

dig +dnssec at DNSKEY

```
 ; «» DiG 9.20.11-1ubuntu2.1-Ubuntu «» +dnssec at DNSKEY
;; global options:  +cmd ;; Got answer:
;; ->HEADER«- opcode:  QUERY, status:  NOERROR, id:  38734
;; flags:  qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version:  0, flags:  do; udp:  65494
;; QUESTION SECTION:
;at.  IN DNSKEY
;; ANSWER SECTION:
at.  3599 IN DNSKEY 257 3 13 cENb...RfsQ==
at.  3599 IN DNSKEY 256 3 13 Yye3...sF/A==
at.  3599 IN DNSKEY 256 3 13 kjE2...Et8w==
at.  3599 IN DNSKEY 257 3 13 2JRF...bX1g==
at.  3599 IN RRSIG DNSKEY 13 1 3600 20260608031708 20260524060119 60960 at.  G1To7+j...I9g==
```

ECDSA Curve P-256 with SHA-256 (algorithm 13) is currently the most widely used signature algorithm in DNSSEC. Note that **.at** uses ECDSA, while **.** (and we have seen this before) uses RSA/SHA-256 (algorithm 8).

Example (aau.at)

dig +dnssec +trace aau.at

```
 aau.at.  600 IN A 143.205.180.80
aau.at.  600 IN NS ns2.aau.at.
aau.at.  600 IN NS ns5.univie.ac.at.
aau.at.  600 IN NS ns10.univie.ac.at.
aau.at.  600 IN NS ns1.aau.at.
;; Received 194 bytes from 192.76.243.2#53(ns10.univie.ac.at) in 97 ms
```

Example (.ac)

Cont.

```
 aau.at.  10800 IN NS ns1.aau.at.
aau.at.  10800 IN NS ns2.aau.at.
aau.at.  10800 IN NS ns5.univie.ac.at.
aau.at.  10800 IN NS ns10.univie.ac.at.
fjscbioio98ccv4od6ka4d7oh5bgrn00.at.  10800 IN NSEC3 1 1 0 - FJSV4TC8DFNIO9SAUK9FQHKH8AAF5POD NS SOA RRSIG DNSKEY
NSEC3PARAM
fjscbioio98ccv4od6ka4d7oh5bgrn00.at.  10800 IN RRSIG NSEC3 13 2 10800 20260609070625 20260527050120 14082 at.
ahgIM1E....CeW1qKDg==
h0js5107nikluu73jqj2tkv9cknnonig.at.  10800 IN NSEC3 1 1 0 - HOKIAJCCNUQ2RRIFRPOA3ED74ONVTD7U NS DS RRSIG
h0js5107nikluu73jqj2tkv9cknnonig.at.  10800 IN RRSIG NSEC3 13 2 10800 20260611085212 20260528050120 14082 at.
ICvqurE....c2qxA==
;; Received 533 bytes from 185.102.12.2#53(u.ns.at) in 84 ms
```

Example (.at)

Cont.

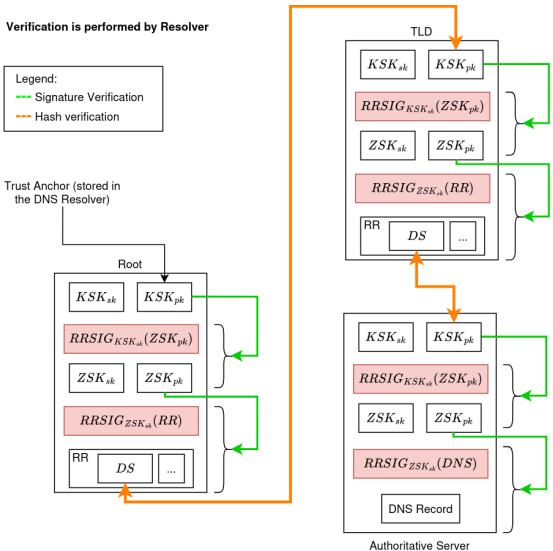
```
 at.  172800 IN NS ns1.univie.ac.at.
at.  172800 IN NS u.ns.at.
at.  172800 IN NS n.ns.at.
at.  172800 IN NS ns2.univie.ac.at.
at.  172800 IN NS j.ns.at.
at.  172800 IN NS r.ns.at.
at.  172800 IN NS d.ns.at.
at.  86400 IN DS 60960 13 2 A00073DD75C499465FE829381CE3F5488FC353D06E68C7F90A04D26A 0D71A694
at.  86400 IN DS 1253 13 2 BA17C1BACB3FB49F7760AD1F7E71E17AB39EE0DF3E9D3BF23FD3D70D 6CF1719E
at.  86400 IN RRSIG DS 8 1 86400 20260612050000 20260530040000 54393 .  FOOhV3gyYc....SFMTpQ==
;; Received 887 bytes from 192.112.36.4#53(g.root-servers.net) in 74 ms
```

Example (.)

Cont.

```
 ; «» DiG 9.20.11-1ubuntu2.1-Ubuntu «» +dnssec +trace aau.at
;; global options:  +cmd
.  305762 IN NS a.root-servers.net.
.  305762 IN NS b.root-servers.net.
.  305762 IN NS c.root-servers.net.
.  305762 IN NS d.root-servers.net.
.  305762 IN NS e.root-servers.net.
.  305762 IN NS f.root-servers.net.
.  305762 IN NS g.root-servers.net.
.  305762 IN NS h.root-servers.net.
.  305762 IN NS i.root-servers.net.
.  305762 IN NS j.root-servers.net.
.  305762 IN NS k.root-servers.net.
.  305762 IN NS l.root-servers.net.
.  305762 IN NS m.root-servers.net.
.  305762 IN RRSIG NS 8 0 518400 20260612050000 20260530040000 54393 .  lzvAzAx21Gn....h8NWBw==
;; Received 1097 bytes from 127.0.0.53#53(127.0.0.53) in 67 ms
```

Chain of Trust (Validation)



Validation Outcomes at Resolver

- **Secure**: signatures validate with a trust chain.
- **Insecure**: unsigned delegation proven (no DS path).
- **Bogus**: should validate, but signatures/proofs fail.
- **Indeterminate**: not enough information to decide.

- Operationally, bogus responses are often treated as SERVFAIL to clients.
- For incident response, spikes in **Bogus/SERVFAIL** often indicate key rollover or signing errors.

NSEC vs NSEC3 for Denial of Existence

- When a resolver queries for a non-existent name, the authoritative server replies with `NXDOMAIN` (Non-eXistent DOMAIN).
- It must provide a cryptographic proof of non-existence to prevent attackers from forging NXDOMAIN responses and spoofing non-existent names.
- **NSEC**: straightforward authenticated denial, but easier zone enumeration and walking.
- **NSEC3**: hashed names; reduces direct zone enumeration.
- Trade-off: NSEC3 can increase computational overhead and complexity.

NSEC

- NSEC (Next Secure) records link existing names in a zone, proving non-existence of queried names.
- Names within a zone are ordered lexicographically; NSEC records point to the next existing name.
- **Example**: if `foo.example.com` and `quo.example.com` exist, an NSEC record might say `foo.example.com NSEC quo.example.com`, proving that any name between them (like `lecture.example.com`) does not exist.
- Vulnerable to zone enumeration: attackers can query for all possible names and collect NSEC records to map the entire zone.

NSEC3

- NSEC3 uses a hash of the domain name instead of the plaintext name, making it more resistant to zone enumeration.
- The hash is computed using a specified algorithm (e.g., SHA-1) and parameters (iterations, salt).
- **Example**: instead of `foo.example.com NSEC quo.example.com`, you might have `HASH(foo.example.com) NSEC3 HASH(quo.example.com)`.
- While it provides better privacy against enumeration, it can increase computational overhead for both authoritative servers and resolvers due to hashing and iteration.

Conclusions and Best Practices

Recent High-Impact DNSSEC DoS

(CVE-2023-50387, disclosed 2024):

- Specially crafted DNSSEC responses can cause extreme CPU consumption in validators.
- Root cause class: expensive combinations of DNSKEY/RRSIG validation work.
- CVSS 3.1 base score commonly listed as 7.5 (High).

Is DNSSEC Enough?

- DNSSEC deployment is necessary for integrity, but not sufficient for resilience.
- Resolver hardening must include:
  - software updates;
  - CPU/memory safeguards;
  - monitoring of validation-failure rates.
- Security controls must cover protocol, software, and operations together.

DNS Privacy Controls in Context

- DNSSEC authenticates data, but does not hide query content.
- Complement with encrypted transport where appropriate:
  - DoT (DNS over TLS)
  - DoH (DNS over HTTPS)
  - DoQ (DNS over QUIC)
- QNAME minimization (RFC 9156) further reduces metadata leakage.

Resolver and Authoritative Hardening Checklist

- Enable DNSSEC validation on recursive resolvers.
- Keep resolver software up to date.
- Disable open recursion; apply ACLs (Access Control Lists) and rate limits.
- Enable response rate limiting on authoritative servers.
- Monitor `SERVFAIL`, `NXDOMAIN` spikes, and unusual query entropy.
- Secure registrar accounts with MFA.
- Audit DS/DNSKEY consistency during every key rollover.

References (Selected)

- RFC 4033/4034/4035: DNSSEC core specifications.
- RFC 6840: DNSSEC clarifications and implementation notes.
- RFC 8901: Multi-signer DNSSEC models.
- RFC 9156: QNAME minimization.
- RFC 9471: glue requirements in referral responses.
- RFC 9619: QDCOUNT constraints for QUERY.
- NVD: CVE-2023-50387 (KeyTrap).
- ISC advisory: CVE-2023-50387 operational impact and fixes.
- Image source: Wikimedia Commons (Domain name space diagram).