DNS message checksums

`draft-muks-dnsop-dns-message-checksums-01`

Mukund Sivaraman

Internet Systems Consortium

IP layer fragmentation

- IP layer PDUs ("packets") can get fragmented — broken into multiple pieces during transmission — because they don't fit in the MTU somewhere on the path.
- Identifier field in the IPv4 header is 16-bit.
- IP fragments present an opportunity for an off-path attacker to succeed in poisoning a reply over IPv4 by injecting spoofed fragments into a network.
- List of problems in *"Fragmentation Considered Poisonous"* by Amir Herzberg and Haya Shulman.
  `http://u.cs.biu.ac.il/~herzbea/security/13-03-frag.pdf`

DNS message poisoning (UDP checksum pass)

- UDP checksum is trivially defeated and is insufficient to protect against malicious modifications of datagrams.
- RFC5452 anti-Kaminsky measures (random source port, random message ID) cannot be used in detecting message modification (poisoning); they can only be used in detecting spoofing at the whole-packet level.
- DNS cookies similarly cannot be used in detecting message modification, except detecting spoofing at the whole-packet level.

Denial of service (UDP checksum fail)

- Response blocking — attacker spoofs bogus fragments causing assembled UDP datagram to fail UDP checksum verification.
- Nameserver blocking — attacker repeatedly blocks responses from a nameserver, and resolver blacklists it denying itself access to the nameserver for all the zones it serves.
- The general problem is that the IP layer is vulnerable as fragment assembly is controlled here. Higher layers may drop assembled packets from incorrect fragments and it's not possible to control fragment assembly from the higher layers, allowing unstoppable disruption to UDP while IP fragmentation is occurring.
- It is worth making better use of `DF=1` and switching to application level fragmentation for larger responses (see DNS message fragments draft) where the application controls assembly.

Off-path attackers

- Unable to snoop on packets (no loss of privacy)
- Able to inject forged packets (poisoning)
- Unable to filter packets (no loss of service)
- Effects: Poisoning of data, some control (protocol information), cause havoc with IP fragments
- Solutions: Strong checks and application level fragmentation with `DF=1` are reasonable protection measures

On-path attackers

- May be able to snoop on packets (loss of privacy)
- May be able to inject forged packets (poisoning)
- May be able to filter packets (loss of service)
- Effects: MITM attacks, TCP RSTs, packet drops, total chaos
- Solutions: Cryptographic signatures (including for control — e.g., TLS cannot stop TCP reset injection), encryption; even then nothing can be done to prevent total loss of service

“Just use DNSSEC!”

- RRSIGs can be used to validate the contents of RRsets.
- Some content of messages do not have RRSIGs and are unprotected, such as NS records and glue at a delegation point in a referral, and EDNS0 options.
- DNSSEC is not designed to be used to validate *what* RRs were chosen by a server to be part of a message.

DNSSEC “*what* RRs” example

- Consider a HTTP service ‘files.example.com’ with address 192.0.2.1 for users in Tolmekia and address 198.51.100.1 for users in Atlantis, which are only accessible from those countries respectively — there is no route to these addresses from outside the respective country.
- NS serves A 192.0.2.1 to Tolmekia, A 198.51.100.1 to Atlantis using views. The zones are signed with the same KSK and ZSK.
- Attacker who’s able to spoof traffic in Atlantis uses shell account in Tolmekia to grab DNS message for Tolmekia and poisons resolver in Atlantis with Tolmekia’s address record with its correct RRSIG.
- User in Atlantis is unable to access ‘files.example.com’ using an address from `AD=1` reply because there’s no route.

“Just use TCP!”

- UDP has annoying problems allowing poisoning, amplification attacks, etc. In DNS, they are mostly known and addressed.
- DNS over TCP doubles roundtrips compared to UDP. Name resolution involves iteration including indirection (glueless NS referral) during lookup. As DNS lookup is at the head of any list of network operations, it increases the turnaround time of every item on the list when resolution is required. This is very conspicuous in parts of the world with large RTTs to the majority of nameservers for domains of medium-to-high popularity and must not be ignored.
- Truncating UDP on purpose to force TCP is worse. It triples roundtrips (1 for UDP attempt, +2 until TCP first data).
- `draft-ietf-dnsop-5966bis` explores ways of improving DNS over TCP, but some problems will take a long time to go away.

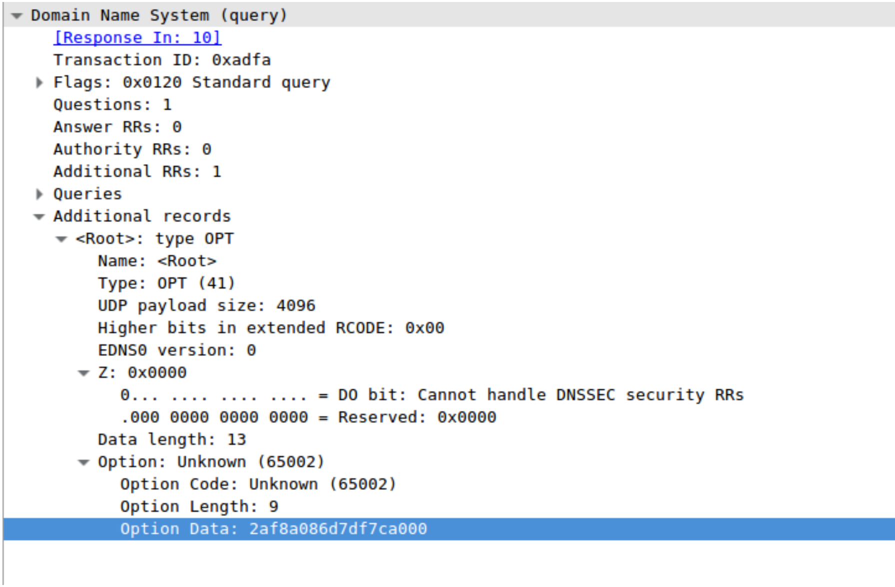
The EDNS CHECKSUM option

- CHECKSUM option tries to protect UDP DNS traffic in its existing form from off-path modifications.
- Option contains 3 fields: NONCE, ALGORITHM and DIGEST

Option field	**Type**	**Field size**
NONCE	byte array	8 octets
ALGORITHM	unsigned integer	1 octet
DIGEST	byte array	variable length

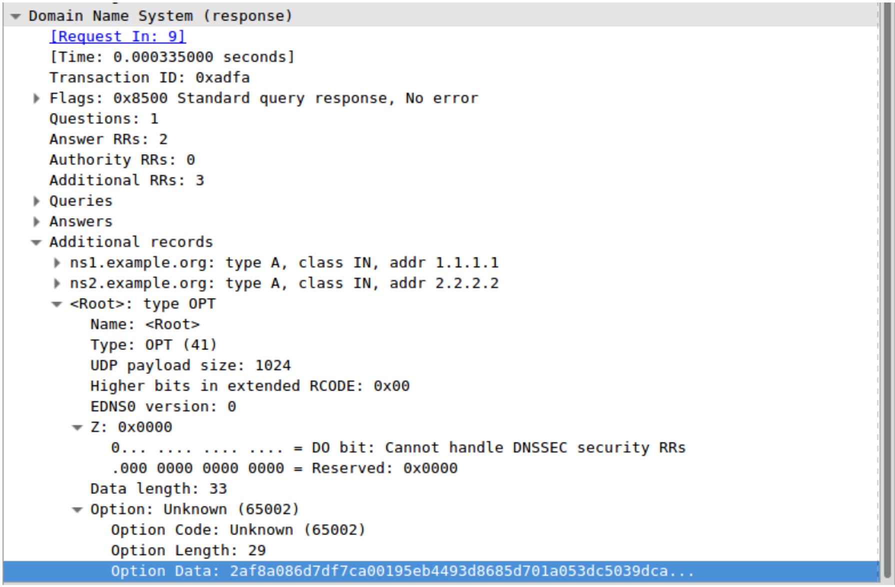
- Client sends DNS query with random NONCE inside an EDNS option.
- Server generates reply message, copies received NONCE to it and computes the checksum digest over the entire message, and returns the DIGEST embedded in the reply to the client.

DNS message with CHECKSUM — `QR=0`



```
▾ Domain Name System (query)
    [Response In: 10]
    Transaction ID: 0xadfa
  ▸ Flags: 0x0120 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▸ Queries
  ▾ Additional records
    ▾ <Root>: type OPT
        Name: <Root>
        Type: OPT (41)
        UDP payload size: 4096
        Higher bits in extended RCODE: 0x00
        EDNS0 version: 0
      ▾ Z: 0x0000
          0... .... .... .... = DO bit: Cannot handle DNSSEC security RRs
          .000 0000 0000 0000 = Reserved: 0x0000
        Data length: 13
      ▾ Option: Unknown (65002)
          Option Code: Unknown (65002)
          Option Length: 9
          Option Data: 2af8a086d7df7ca000
```

DNS message with CHECKSUM — `QR=1`



```
▾ Domain Name System (response)
    [Request In: 9]
    [Time: 0.000335000 seconds]
    Transaction ID: 0xadfa
  ▸ Flags: 0x8500 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 3
  ▸ Queries
  ▸ Answers
  ▾ Additional records
    ▸ ns1.example.org: type A, class IN, addr 1.1.1.1
    ▸ ns2.example.org: type A, class IN, addr 2.2.2.2
    ▾ <Root>: type OPT
        Name: <Root>
        Type: OPT (41)
        UDP payload size: 1024
        Higher bits in extended RCODE: 0x00
        EDNS0 version: 0
      ▾ Z: 0x0000
          0... .... .... .... = DO bit: Cannot handle DNSSEC security RRs
          .000 0000 0000 0000 = Reserved: 0x0000
        Data length: 33
      ▾ Option: Unknown (65002)
          Option Code: Unknown (65002)
          Option Length: 29
          Option Data: 2af8a086d7df7ca00195eb4493d8685d701a053dc5039dca...
```

Security considerations

- Risk of downgrade attack by IP fragment spoofing — client can detect it and retry over TCP.
- CHECKSUM cannot protect against UDP checksum validation failures due to spoofed IP fragments, allowing response blocking and NS blocking attacks. DNS message fragments can be used in addressing these problems.
- On-path message poisoning is better handled by DPRIVE; would require significantly different operation and possible protocol changes.
- Checksum digest computation has a processing overhead.
- Nonces require a pseudo-random entropy source.

Open issues

- Whether to replace the NONCE field with dependency on DNS cookies — after implementation experience, this doesn't seem so good as the packet overhead of the NONCE field is very small, verifying the NONCE takes a single instruction, and there is a larger overhead to checking that the cookie option was found.
- Whether to merge checksums into DNS cookies — checksums have a computation overhead that may not be welcome in cookies. Other than that, we can try this.

100 END

Thanks for watching.

A BIND implementation is in the `dns-message-checksums` branch at: `https://github.com/muks/bind9/`