Greasing Protocol Extension
Points in the DNS

Shumon Huque & Mark Andrews
March 18th 2024
DNS Operations Working Group
Internet Engineering Task Force (IETF) 118 Meeting
Brisbane, Australia
Greasing - technique that exercises the regular use of unallocated protocol code points
- to prevent ossification of their current usage patterns by middleboxes and/or deficient DNS actors.

Already used successfully in TLS and QUIC
- RFC 8701: Applying GREASE to TLS Extensibility

Also see IETF’s EDM (Evolvability, Deployability, & Maintainability) program

This draft describes the application of this technique to the DNS protocol
- have resolvers periodically send out queries with unallocated code points, and collect the results
What protocol elements could we grease?

- New DNS header flags
- New EDNS header flags
- New Opcodes
- New EDNS options
- New Resource Record types
  - Of various subtypes: data, meta, and q-types
- New Resource Record classes
- Transports: TCP, TLS, QUIC (maybe some of it covered by RFC 9539)
How to choose unallocated code points?

- At random from the entire range
- From a reserved range for greasing
<table>
<thead>
<tr>
<th>Protocol Element</th>
<th>Size</th>
<th># of Values (# used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Header Flags</td>
<td>7-bits</td>
<td>7 (6)</td>
</tr>
<tr>
<td>EDNS Header Flags</td>
<td>16-bits</td>
<td>16 (1)</td>
</tr>
<tr>
<td>Resource Record Type</td>
<td>16-bits</td>
<td>65,536 (~90)</td>
</tr>
<tr>
<td>Opcode</td>
<td>4-bits</td>
<td>16 (6)</td>
</tr>
<tr>
<td>EDNS Version</td>
<td>8-bits</td>
<td>256 (1)</td>
</tr>
<tr>
<td>EDNS Opt Code</td>
<td>16-bits</td>
<td>65,536 (19)</td>
</tr>
<tr>
<td>RR Class</td>
<td>16-bits</td>
<td>65,536 (7)</td>
</tr>
</tbody>
</table>

(Note: distinction between fields that represent range of values vs set of discrete flags)
RR type space organization

- Data Types
- Meta and Q-Types
- Private Use
- Reserved ranges
- etc.
6. Sampled Selection of Traffic

To avoid the overhead of needing to retry many queries in the event of large scale intolerance of unallocated code points, only a sampled fraction of DNS requests emitted by a resolver should advertise unallocated code points. Many DNS resolvers are very high transaction rate systems, so only a small sample size of such DNS requests is sufficient to get a rough picture of non-compliant servers, perhaps 1 in 1000 requests? Furthermore, a community effort of aggregating and analysing the results of failed queries from many DNS resolver operators can provide an even more comprehensive view of the ecosystem.
Failure detection & Response

- Error response where there should not have been an error.
- Failure to give expected error (e.g. EDNS BADVERS)
- Failure to respond (RFC 8906)
  - Need to distinguish from unavailable servers
  - How? sending additional queries to same server?

- Retry without unallocated codepoint
- Except for when a new query has been constructed (e.g. with a greased RR type or RR class) - log error and move on.
Telemetry & Analysis

- Individual server logs?
- Proactive alerting to authority operator (DNS Error Reporting?)
- Centralized collection and analysis (DNS-OARC)?
  - Anonymization of some parameters will likely be needed
Open Discussion