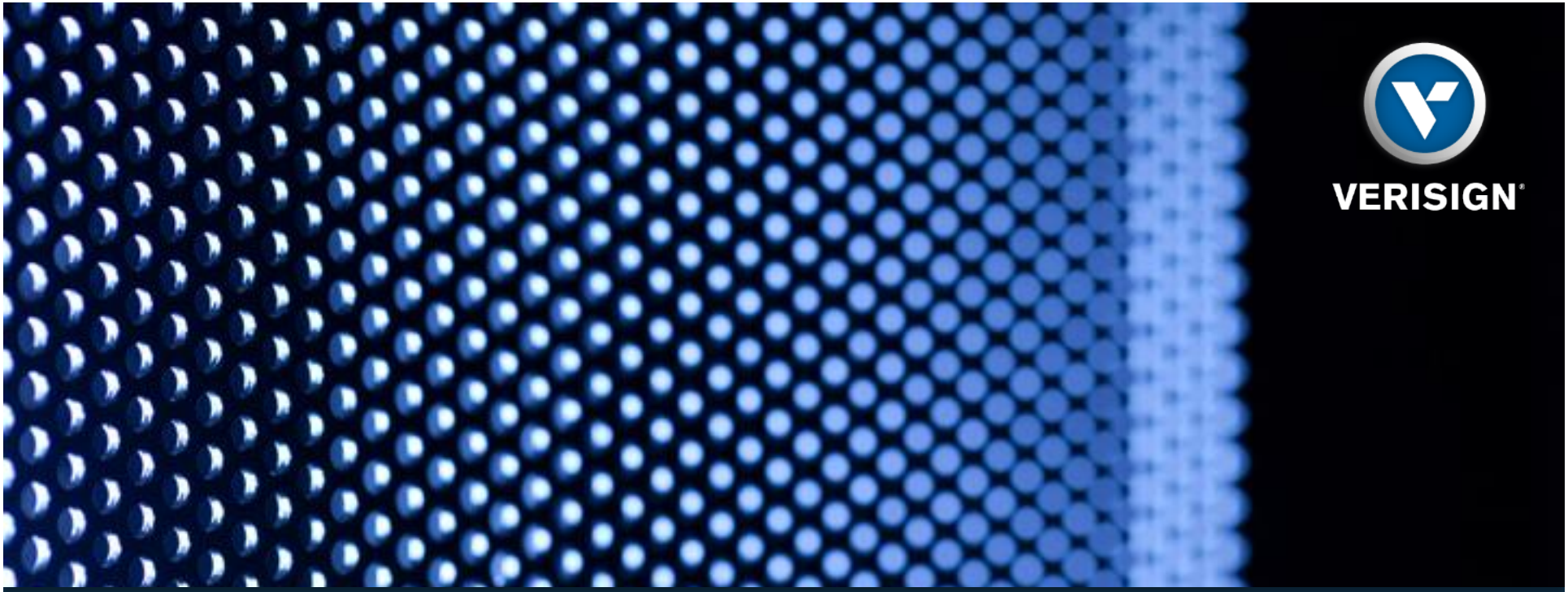


VERISIGN

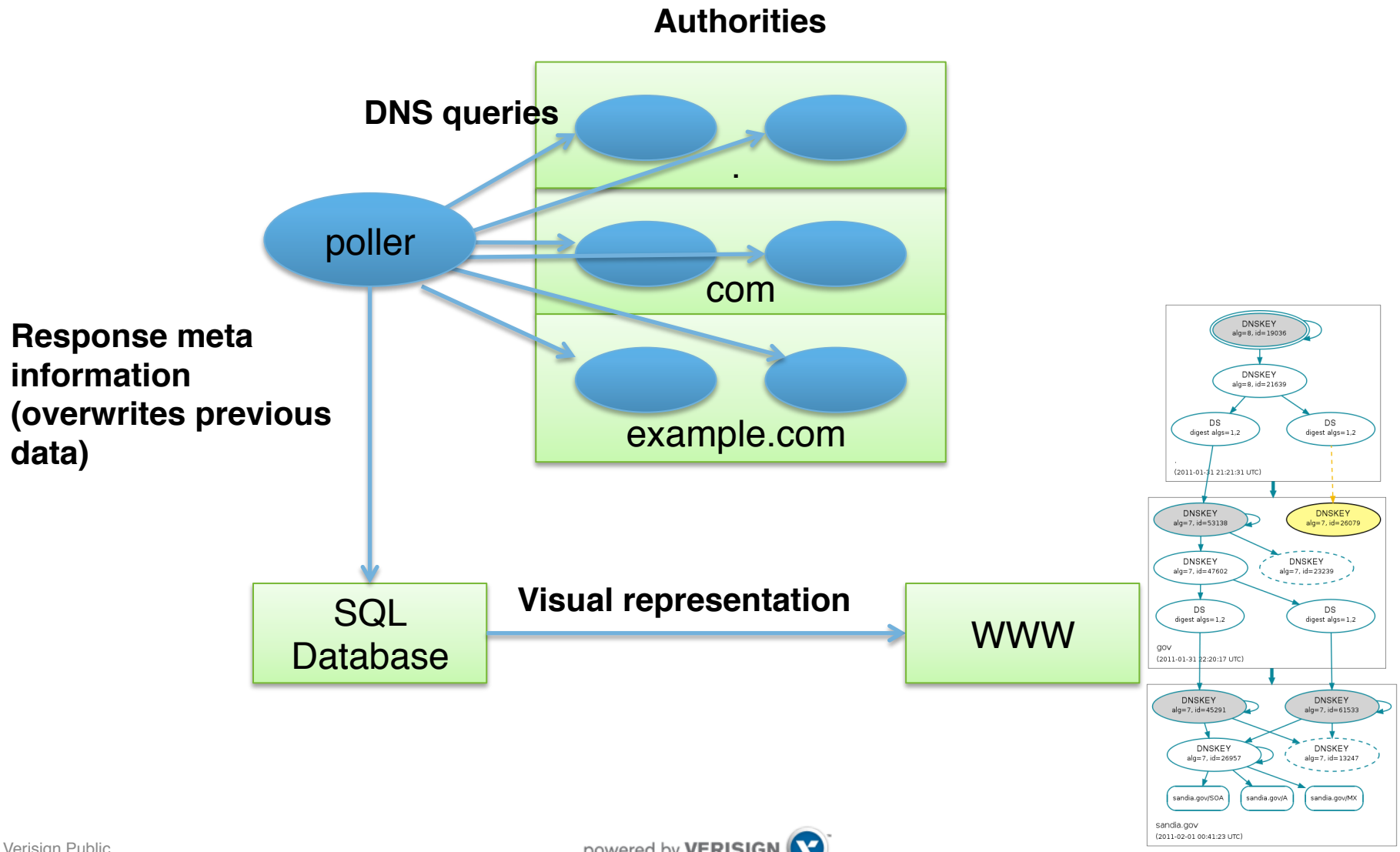
Portable DNS Analysis

Casey Deccio, Verisign Labs

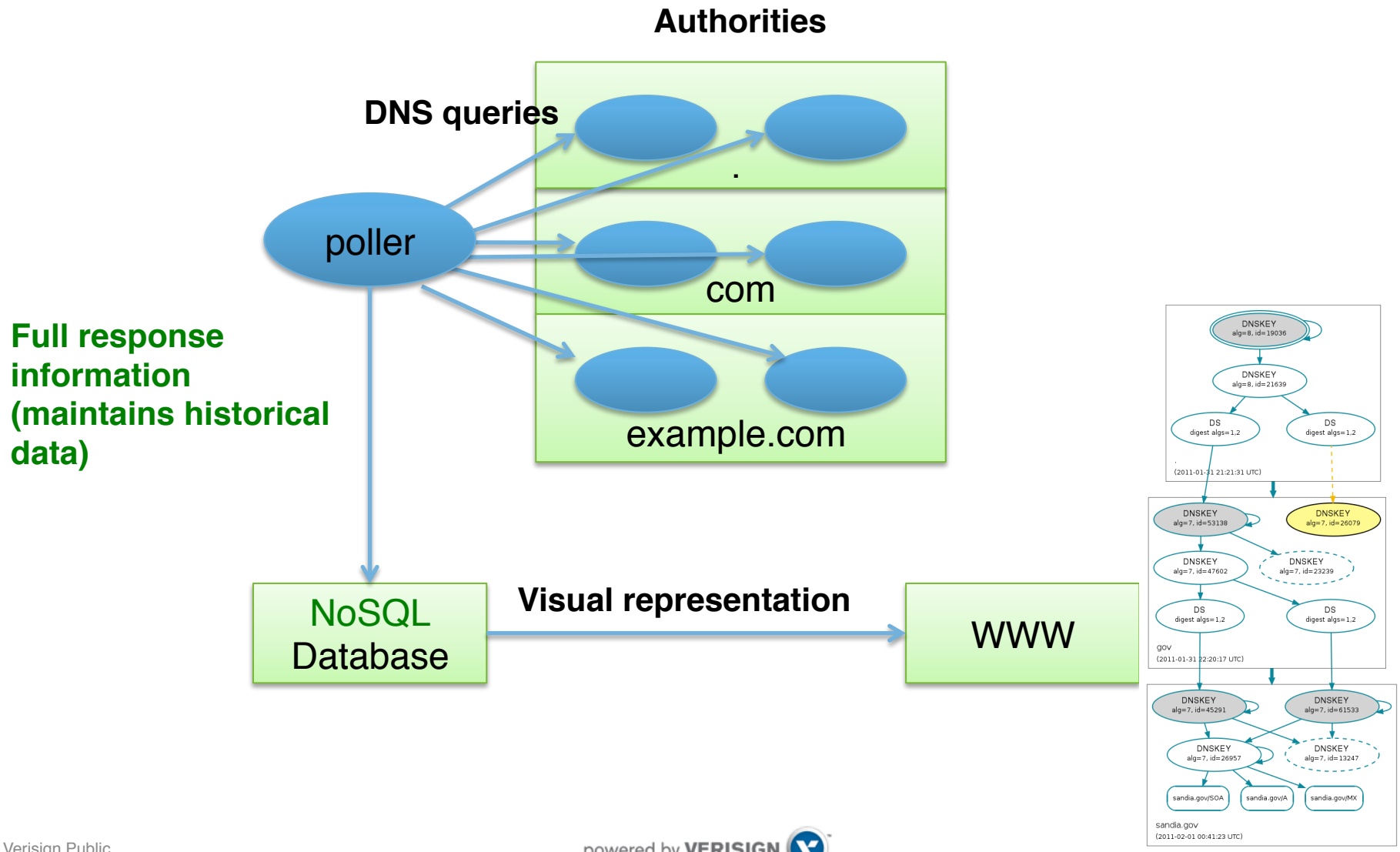
DNS-OARC Spring 2014 Workshop

May 10, 2014, Warsaw, Poland

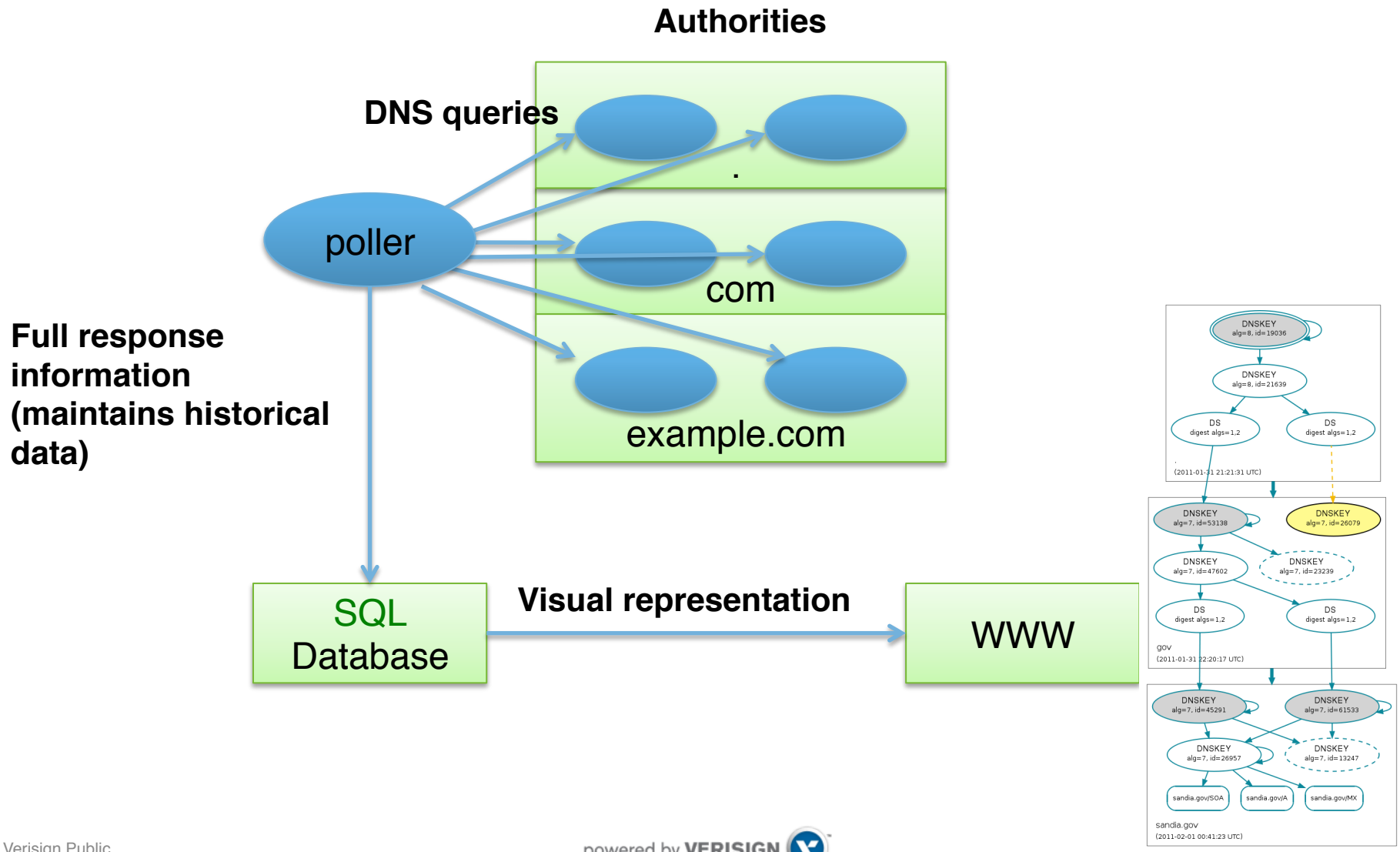
DNSViz monitoring – Generation 1



DNSViz monitoring – Generation 2

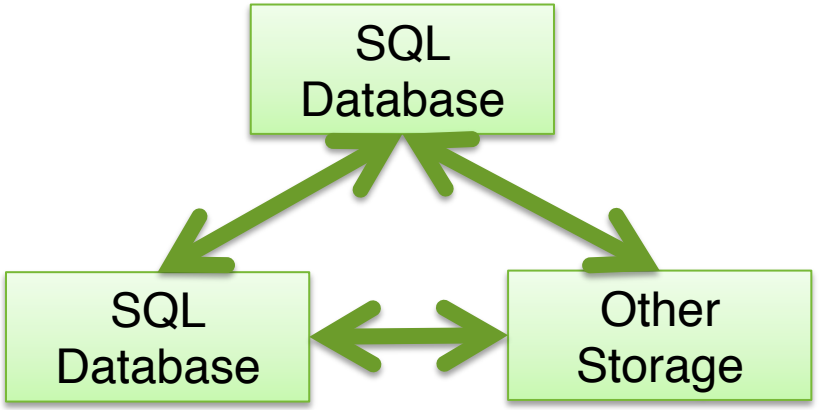


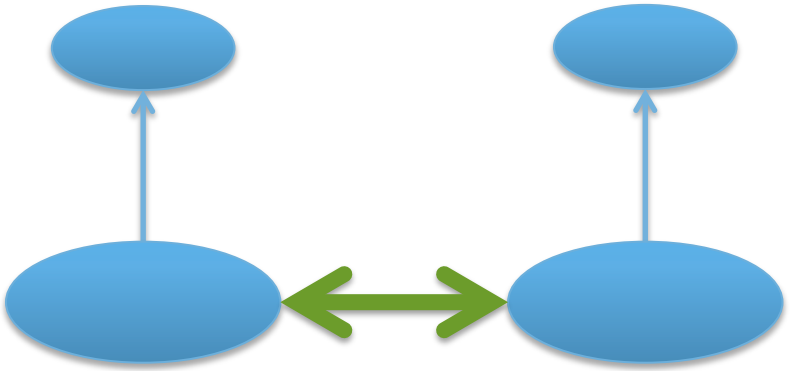
DNSViz monitoring – Generation 3



Challenges - Portability

- Across storage backends
- Across analysis tools
  - DNSViz
  - DNSSEC Debugger
  - Zonecheck
  - DNSCheck
  - dig
  - drill
  - etc…
- Across vantage points



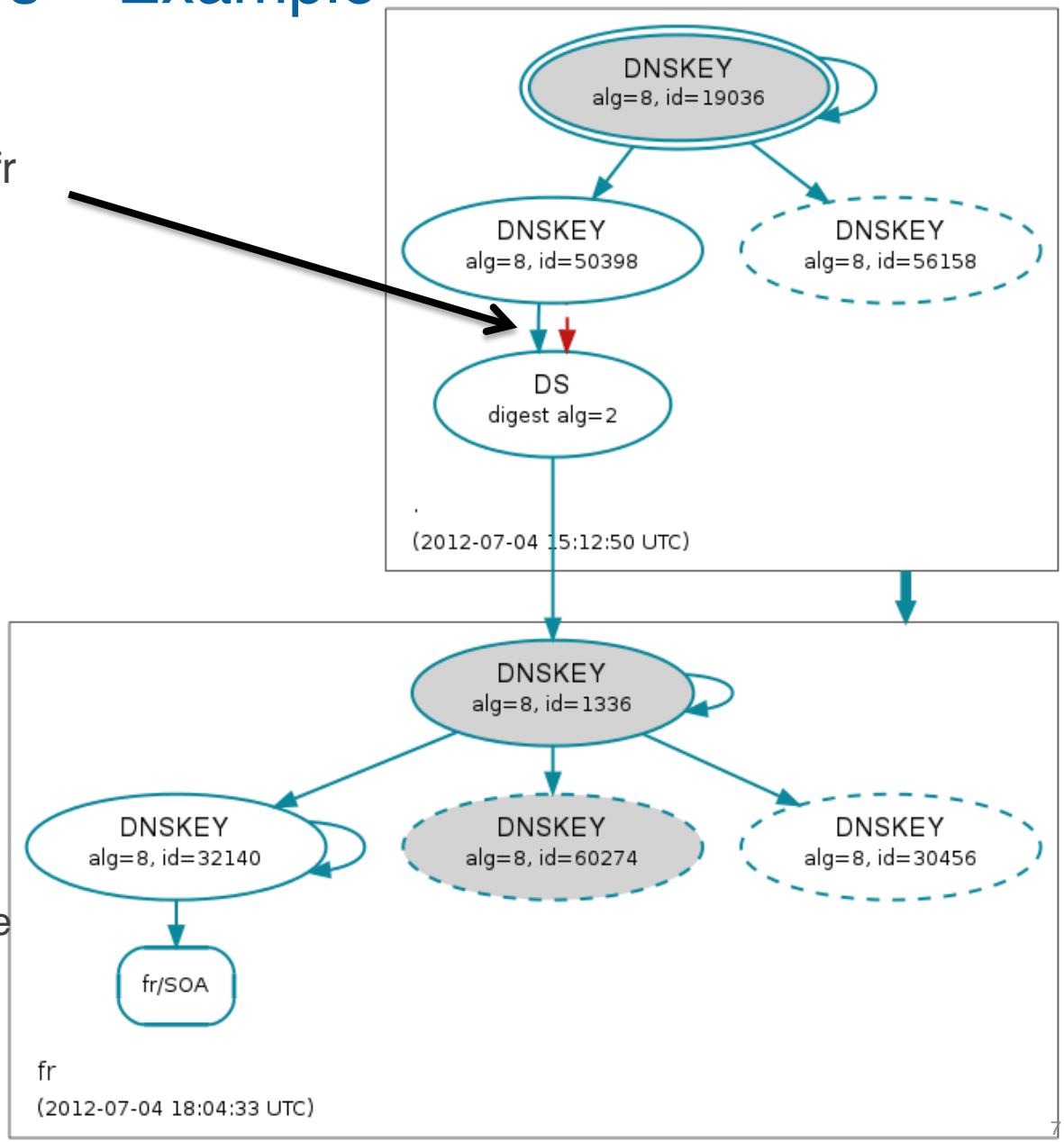


Diagnostic Queries

- More than just a query/response
- Some servers are unresponsive (or respond incorrectly) with “default” query options
- Extra effort is sometimes required to get a response or diagnose problems
- Example symptoms:
  - Timeout
  - Form error
  - Bad RCODE, e.g., SERVFAIL, NOTIMP
- Example diagnostics:
  - Reduce UDP max payload size
  - Disable EDNS

Diagnostic Queries – Example

- Two root server instances didn't return RRSIGs for .fr DS records
- Known:
  - No EDNS in response
- Unknown:
  - EDNS in request?
- Possible causes
  - Servers not responding properly with EDNS (and therefore RRSIGs)?
  - Middle boxes dropping EDNS queries?
  - Servers/links busy (hence the timeouts)?



Portable DNS Query Responses

- Query response information
  - Initial request options
    - Flags
    - EDNS: version, UDP max payload, flags, options
    - TCP
  - History of responses and changes to request options
  - Wire message (base64 encoded)
- Representation
  - JSON

Serialized Query Response – Simple Example


```
{
   "qname": "example.com.",
   "qclass": “IN”,
   "qtype": “A”,
   "flags": 256,
   "edns_version": 0,
   "edns_max_udp_payload": 4096,
   "edns_flags": 32768,
   "edns_options": [],
   "responses": {
      "8.8.8.8": {
         "192.168.1.8": {
            "message":
"pUeBoAABAAIAAAABB2V4YW1wbGUDY29tAAABAAHADAABAAEAAEBJAARduNh3wAwALgABAABAS
QCfAAEIAgABUYBTdHerU2rBfBknB2V4YW1wbGUDY29tAC1qOK609O1BVxvRBZ5vs5bRSJVdN0niY17
w+cZ7xBjH/3nLUWZb1sKUCE+hVuGg9mdimrw2JOhMkh3S7ALnNM2GHSpas9M1RnKN/
aqenYD4m03ag6wNk/E
+PSqrMGq4/8PYxqKXYQLlwH4sbT44rnddIGdEVHFe50tNg7lBlep8AAApAgAAAIAAAAA=",
            "tcp_first": false,
            "response_time": 0.022,
            "history": []
         }
      }
   }
}
```

Serialized Query Response – Timeout Example

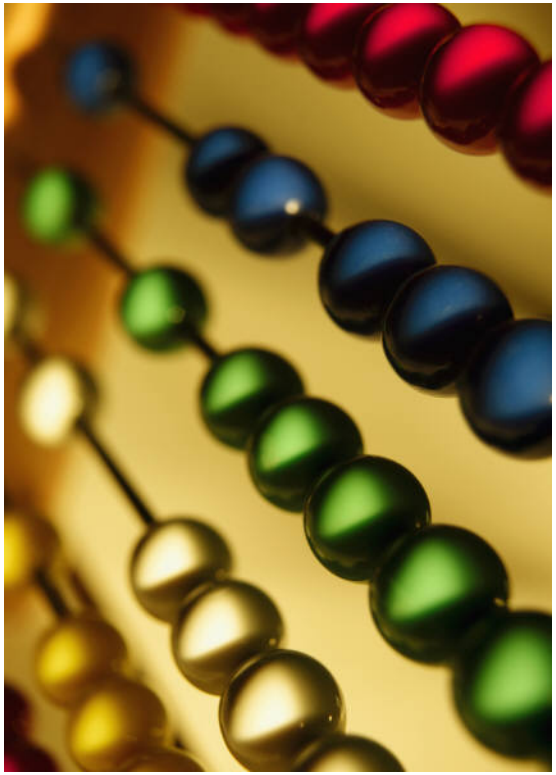

```
{
    "qname": "example.com.",
    "qclass": “IN”,
    "qtype": “A”,
    "flags": 256,
    "edns_version": 0,
    "edns_max_udp_payload": 4096,
    "edns_flags": 32768,
    "edns_options": [],
    "responses": {
        "8.8.8.8": {
            "192.168.1.8": {
                "message": null,
                "error": "TIMEOUT",
                "tcp_first": false,
                "response_time": 2.994,
                "history": [
                    {
                        "response_time": 3.001,
                        "cause": "TIMEOUT",
                        "action": "NO_CHANGE",
                    },
```


(cont’d)


```
                    {
                        "response_time": 3.001,
                        "cause": "TIMEOUT",
                        "action": "CHANGE_UDP_MAX_PAYLOAD",
                        "action_arg": 512
                    },
                    {
                        "response_time": 3.001,
                        "cause": "TIMEOUT",
                        "action": "NO_CHANGE",
                    },
                    {
                        "response_time": 3.001,
                        "cause": "TIMEOUT",
                        "action": "DISABLE_EDNS",
                    }
                ]
            }
        }
    }
}
```

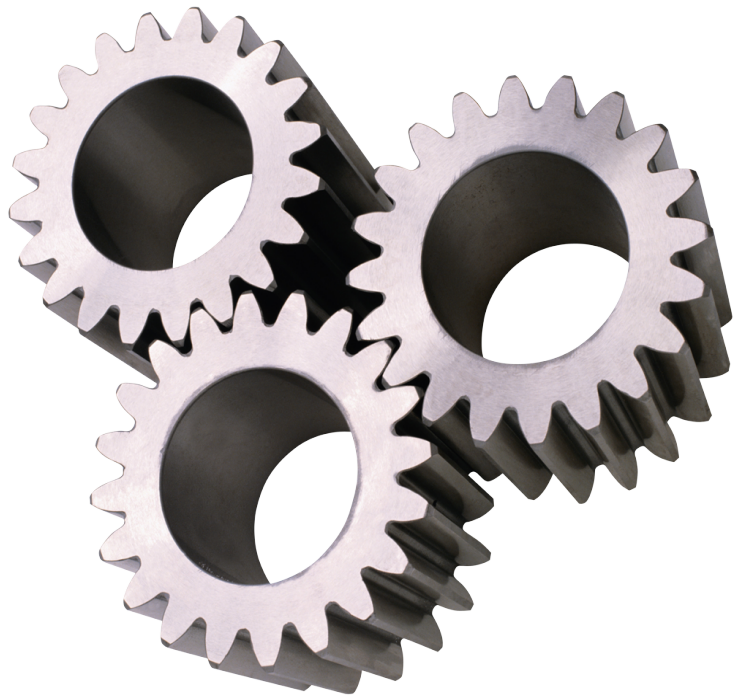
Portable DNS Query Responses – options

- Message granularity options:
  - Actual QueryID vs. derived or 0
  - RRset ordering within section
  - RData order preservation within RRset
- History detail options:
  - Full wire of full message requiring modified re-query
  - Message size (if any) of full message requiring modified re-query
- Option tradeoffs:
  - Low level inspection (name compression, malformed messages, etc.)
  - Multiple message consolidation (i.e., message comparison)

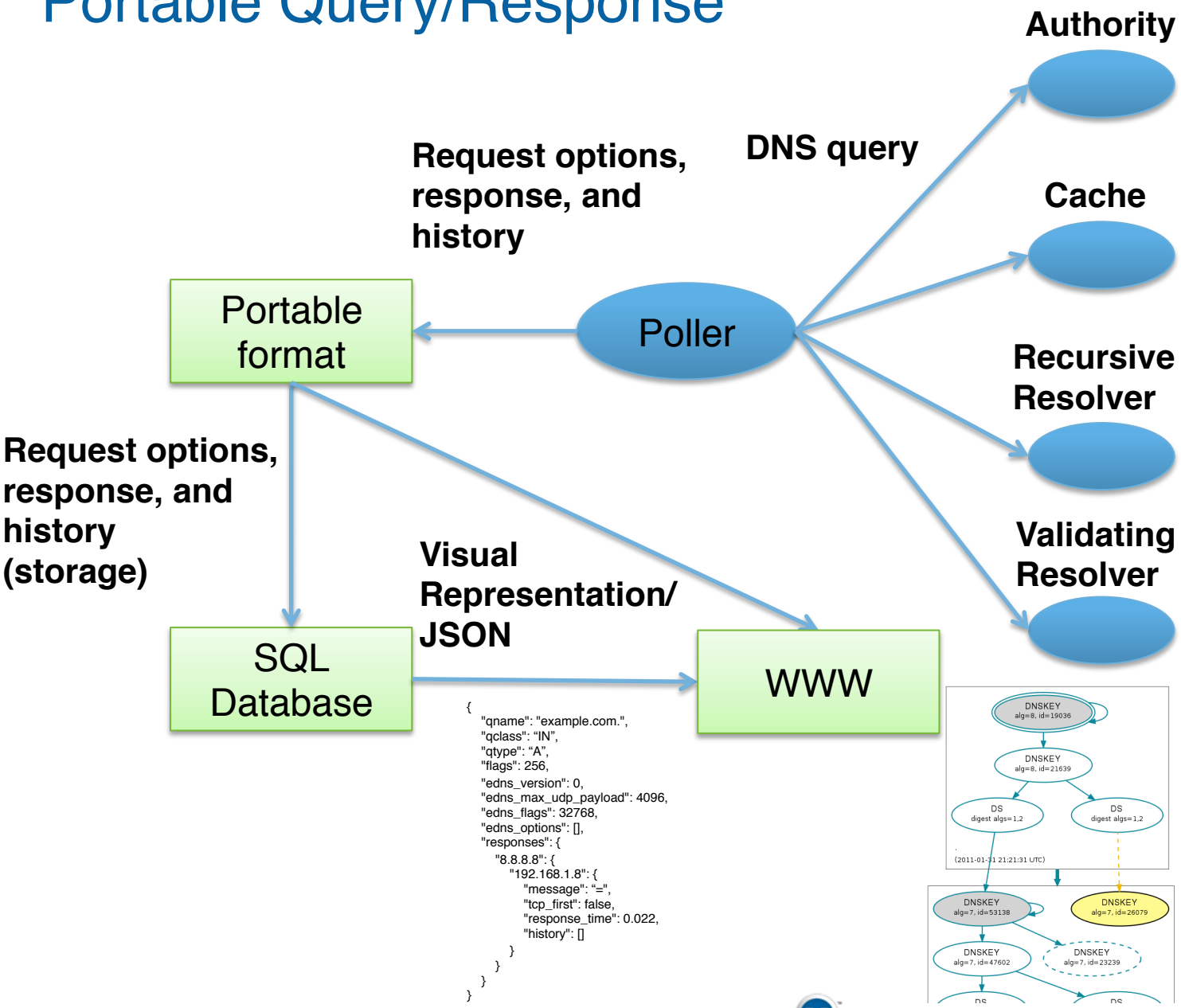


Parameterizing DNS Requests

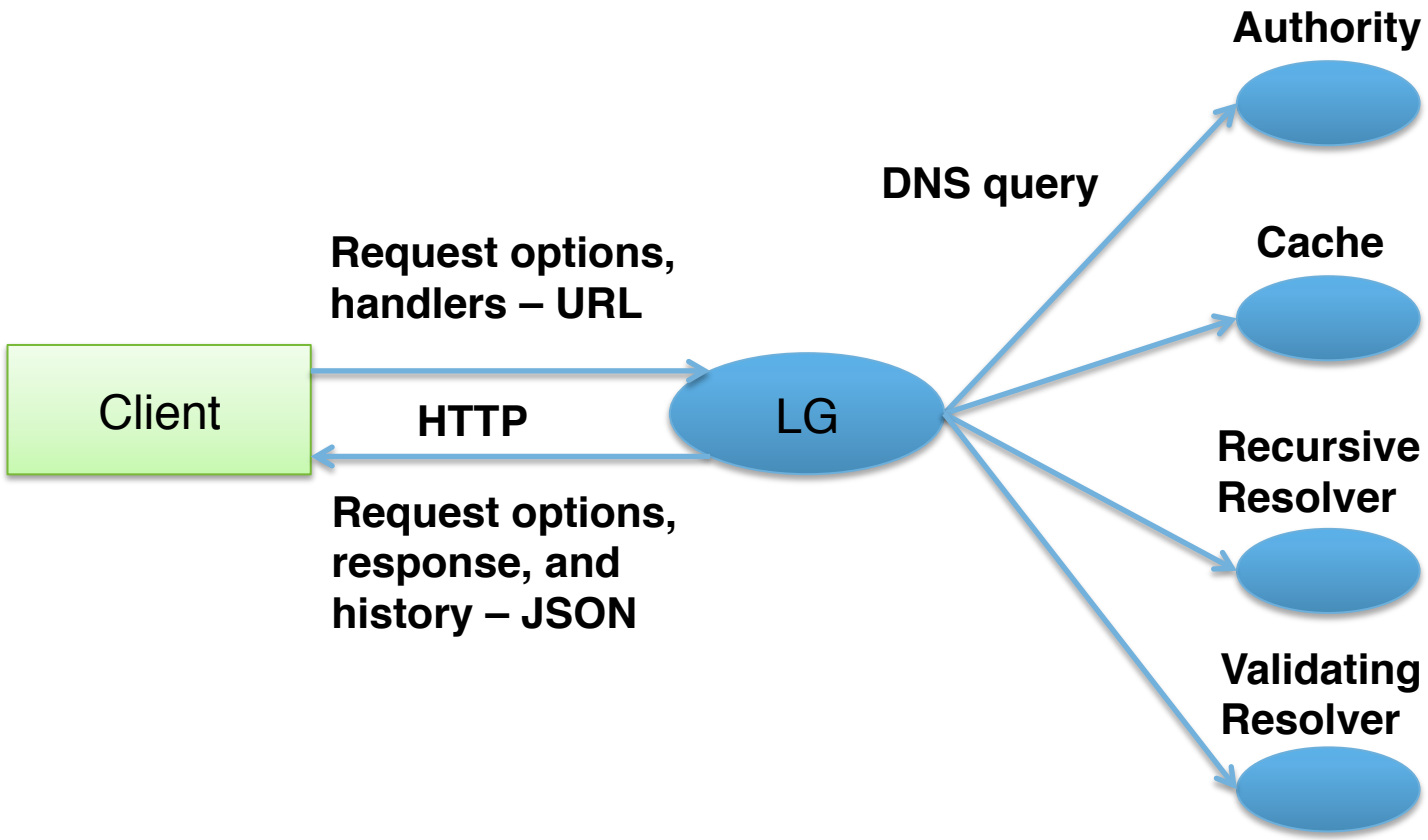
- Initial options
  - Flags
  - EDNS: version, UDP max payload, flags, options
  - TCP
  - Timeout
- Requests handlers
  - Based on a response (or error or timeout), specify action to take



Portable Query/Response



Parameterized Looking Glass



DNSViz (G4) – Response Handler

```
class DNSResponseHandler(object):
    '''A base class for handling DNS responses (or exceptions)
    arising from a query attempt.'''

    def handle(self, response_wire, response, response_time):
        raise NotImplemented
```

- handle() method determines whether to:
  - Retry, optionally with modified options
  - Pass it to the next handler; or
  - Stop immediately, bypassing all future handlers

DNSViz (G4) – Response Handler Example

```
class UseTCPOnTCFlagHandler(DNSResponseHandler):
    '''Retry with TCP if the TC flag is set in the response.'''

    def handle(self, response_wire, response, response_time):
        if response_wire is not None and \
                ord(response_wire[2]) & 0x02:
            self._params['tcp'] = True
            return DNSQueryRetryAttempt(response_time, \
                    RETRY_CAUSE_TC_SET, len(response_wire), \
                    RETRY_ACTION_USE_TCP, None)
```

- Retries using TCP if TC bit is set in the message

DNSViz (G4) – Response Handler Example

```
class ReduceUDPMaxPayloadOnTimeoutHandler(DNSResponseHandler):
    '''Reduce the EDNS UDP max payload after a given number of
    timeouts.  Some servers attempt to send payloads that exceed
    their PMTU.'''

    def __init__(self, reduced_payload, timeouts):
        self._reduced_payload = reduced_payload
        self._timeouts = timeouts

    def handle(self, response_wire, response, response_time):
        timeouts = self._get_num_timeouts(response)
        if timeouts >= self._timeouts and \
                self._request.payload > self._reduced_payload:
            self._request.payload = self._reduced_payload
            return DNSQueryRetryAttempt(response_time,
                        RETRY_CAUSE_TIMEOUT, None,
                        RETRY_ACTION_CHANGE_UDP_MAX_PAYLOAD,
                        self._reduced_payload)
```

Initialize timeout threshold and reduced UDP max payload value

Handle appropriately

DNSViz (G4) – DNSQueryFactory

```
class DNSQueryFactory(object):
    '''A simple, extensible class interface for instantiating
    DNSQuery objects.'''

    flags = 0
    edns = -1
    edns_max_udp_payload = 4096
    edns_flags = 0
    edns_options = []

    query_timeout = 3.0
    max_attempts = 5
    lifetime = 15.0

    response_handlers = []
```

Options are class variables, the defaults overridden in child classes.

DNSViz (G4) – DNSQueryFactory Examples

```
class SimpleDNSQuery(DNSQueryFactory):
    '''A simple query, no frills.'''

    pass

class RecursiveDNSQuery(SimpleDNSQuery):
    '''A simple recursive query.'''

    flags = SimpleDNSQuery.flags | dns.flags.RD

class StandardQuery(SimpleDNSQuery):
    '''A standard old-school DNS query that handles truncated
    packets.'''

    response_handlers = SimpleDNSQuery.response_handlers +
        [UseTCPOnTCFlagHandler()]
```

Portable DNS Analysis Components

- Multiple, portable DNS queries
  - Queries for desired record(s) (e.g., example.com/A)
  - Queries for dependencies (e.g., CNAME, MX, etc.)
  - Queries to build chain of trust – DS/DNSKEY from ancestries
- Additional analysis options
  - Comprehensive (all authoritative servers queried) vs. any server
  - Authoritative vs. cache
  - Inclusion of delegation boundaries (includes queries to determine delegation points)

Future Work

- Formalize standard for portable DNS query and analysis
  - e.g., IETF
- Release new DNSViz (G4) analysis framework and tools for general use
  - DNSViz – command line
  - DNSViz – Web interface

powered by

VERISIGN™