getdns – a modern, open source DNS API implementation

Verisign Labs and NLnet Labs
https://getdnsapi.net

ccNSO TechDay, ICANN49
March 24, 2014

Introduction

- We implemented a new DNS API – what can this be used for?
  - General OS stub resolver.
  - Recursive resolution.
  - DNS resolver built into browser.
  - DNS resolver built into other apps and even servers.
- What API spec did we implement?
  - Community design by applications users needing more powerful DNS – spec published under Creative Commons CCA license, edited by Paul Hoffman, available at http://vpnc.org/getdns-api
  - DNS experts gave input and review but did not drive.
  - Goal was satisfaction of DNS user/app needs (roughly in order): standard asynchronous capability, multi-platform portability, easy upgrade to new DNS features, usability of DANE, easy general use of DNSSEC.
  - Current spec is an update capturing our implementer findings.

Make it easier for non-DNS experts to access the powerful features and evolving capabilities of DNS

Quick Info

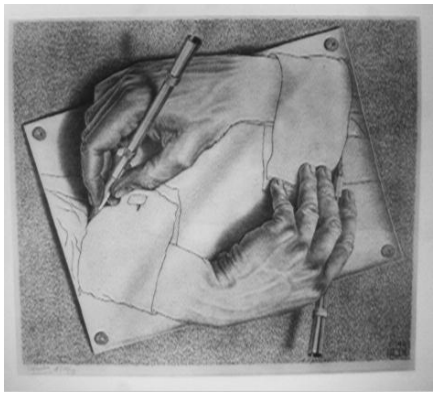
- Verisign Labs in partnership with NLnet Labs
  - Strong team of Open Source developers, QA
- The implementation is open source under New BSD license
- We released a low-key version 0.1.0 timed with IETF 89 (2-7 Mar, 2014, London)
- Release and issue management on github:
  - https://github.com/getdnsapi/getdns
- Website offering links, as well as binaries, documentation:
  - https://getdnsapi.net

Some major features

- Works with a variety of event loops, each built as a separate shared library
  - Details in wiki of the github repo
  - libevent
  - libev
  - libuv
- Full DNSSEC support - base is Unbound, from NLnet Labs
- Attention to IDN handling
- Platforms as of now
  - Linux (RHEL/CentOS), MacOS, FreeBSD, iOS (now rough but usable)
  - Windows, Android later in the year

One API, two modes

- Stub resolver
  - Often implemented via local library (e.g. libresolv)
  - Provides entry points for applications (e.g. gethostbyname)
  - Relies on a recursive name server
  - May not cache, but may implement e.g. single local cache
- Recursive Resolver
  - Typically receives DNS requests via wire protocol
  - Iterates on behalf of clients or applications
  - Typically leverages caching



- getdns-api context controls which of these (2 modes)
- When DNSSEC is enabled for stub mode, the stub can iterate just DNSSEC validation on its own behalf

DNSSEC in the API and implementation

- DNSSEC validation is off by default for stub mode (by design group consensus), but easy to turn on – use of extensions defined in API
  - dnssec_return_status
  - dnssec_return_only_secure
  - dnssec_return_validation_chain

- The API spec allows enabling DNSSEC on a per-request basis via setting the dnssec_return_status extension. For convenience, the implementation provides a means to enable this extension for every request in a given context
  - Documented in getdnsapi repo community wiki

API examples – getdns_general()

- This API is not familiar to C/systems/DNS programmers, but it matches style of web/apps programmers
- getdns_general is typical of public entry points
- Handle arbitrary resource record types

```
getdns_return_t
getdns_general(
  getdns_context_t       context,
  const char             *name,
  uint16_t               request_type,
  struct getdns_dict     *extensions,
  void                   *userarg,
  getdns_transaction_t   *transaction_id,
  getdns_callback_t      callbackfn
);
```

API examples - getdns_address()

- Handles requests by host name
- Always returns both IPv4 and IPv6 addresses
- Uses all name spaces from the context

```
getdns_return_t
getdns_address(
  getdns_context_t       context,
  const char             *name,
  struct getdns_dict     *extensions,
  void                   *userarg,
  getdns_transaction_t   *transaction_id,
  getdns_callback_t      callbackfn
);
```

API examples - getdns_hostname()

- Accepts either IPv4 or IPv6 address

```
getdns_return_t
getdns_hostname(
  getdns_context_t       context,
  struct getdns_dict     *address,
  struct getdns_dict     *extensions,
  void                   *userarg,
  getdns_transaction_t   *transaction_id,
  getdns_callback_t      callbackfn
);
```

API examples - getdns_service()

- Returns relevant SRV information

```
getdns_return_t
getdns_service(
  getdns_context_t       context,
  const char             *name,
  struct getdns_dict     *extensions,
  void                   *userarg,
  getdns_transaction_t   *transaction_id,
  getdns_callback_t      callbackfn
);
```

Reaction and Plans

- Immediate interest from open source and IETF communities following the 0.1.0 release
  - Multiple forks of the getdns and getdns-ios repos already
  - Received a patch for RHEL packaging the day of release
  - Mac Homebrew formula has appeared
  - Supportive statements at IETF from folks from Microsoft, Akamai, Facebook, Cisco, Mozilla, ...
  - Particular interest in Node.js and Python bindings
- Release 0.1.1 that comes out this week will include Node.js bindings; Python bindings due around 11 Apr
- Upcoming trials in web developers' hackathon (at TNW)

Still to be implemented

- These items are in the API spec and will appear in subsequent release
  - MDNS and NetBIOS namespaces – included in spec
  - DNS search suffixes – getdns_context_set_append_name, getdns_context_set_suffix – following DNSOP discussions…
  - GETDNS_TRANSPORT_TCP_ONLY_KEEP_CONNECTIONS_OPEN
  - Full set of EDNS(0) and OPT extensions
  - …
  - Full list in README

Implementation Detail - Dependencies

- Are linked outside the build tree, with configure finding them
- We strive to minimize them
- Current set
  - libldns and libunbound from Nlnet Labs (libldns requires openssl headers and libraries)
  - libexpat
  - libidn from FSF, version 1

The getdns core team

- Craig Despaux
- Angelique Finan
- Neel Goyal
- Olaf Kolkman
- Allison Mankin
- Melinda Shore
- Willem Toorop
- Duane Wessels
- Wouter Wijngaards
- Glen Wiley

Questions?

Most answers will be found at

getdnsapi.net

and

github.com/getdnsapi