Private DNS

Phill Hallam-Baker

Design Origin

- DNSSEC
  - Limited to authenticity of authoritative DNS
  - Requires a clean port 53 for client/resolver
- DNSCurve
  - Transport layer security to Authoritative DNS
  - Public key in the client/server transaction loop
    - DDoS potential
  - Assumes non-DNS syntax works on port 53

Port 53 Interference

- Stupidity
  - Limit DNS packets to 500/512 bytes
  - Disable TCP fallback
  - Strip out unknown RRs
- Malice
  - Redirect all DNS traffic to own service
  - Strip out ‘undesirable’ RRs

Observation

Privacy requirements assume malice.

Objectives

- 100% Connectivity
- Performance equal or better than existing
- Stateless transactions with no public key
- Bypass interference
- Eliminate amplification and relay attacks
- Low footprint, complexity
- Enable curated DNS

- (Confidentiality)

Approach

- RP (User or Enterprise) chooses service
  - Binds each device to their chosen service
  - Devices only use the nominated service*

* Except for bootstrap situations, e.g. WiFi

Protocol Architecture

- Key Agreement / Device Binding
  - Negotiate crypto parameters
  - Agree shared secret and session ID
  - Specify IP addresses, services for Hosts

- Service Transports
  - UDP – works for 97% of cases
  - Web Service – guarantees connectivity

Enterprise User

- Alice is new hire at Example corp
  - Wants to access corporate net from his mobile
  - IT gives him
    - Connection service domain name [example.com]
    - One time passcode [wej2i-h23io-d209d-jeiqi]
  - Alice enters above into device
  - Device negotiates connection:
    - IP addresses, session keys, protocols of Alice's hosts
    - Session Identifier (opaque)

Casual User

- Bobis somewhat privacy conscious
  - Enters public DNS provider [pdns.comodo.com]
- Provider does not require authentication
  - But can link all Bob’s traffic by session ID

Privacy Sensitive User

- Carol is very privacy conscious
  – Does not want public provider to track her
  – Enters multiple public DNS providers
  – Checks the ‘renegotiate’ option in device

- Device pre-negotiates multiple sessions
  – Caches pre-negotiated sessions for future use.
  – Provides privacy but loses other protections

Connection Service Example Response


```
HTTP/1.1 OK Success
 Content-Length: 578
 Date: Fri, 09 May 2014 20:58:44 GMT
 Server: Microsoft-HTTPAPI/2.0

 {
  "TicketResponse": {
   "Status": 200,
   "StatusDescription": "Success",
   "Cryptographic": [],
   "Service": [{
     "Service": "private-dns-resolver",
     "Name": "localhost",
     "Port": 9090,
     "Priority": 100,
     "Weight": 100,
     "Transport": "UDP",
     "Cryptographic": {
      "Secret": "
 SwVyt3p_tkMeneeYtqnw5g",
      "Encryption": "A128CBC",
      "Authentication": "HS256T128",
      "Ticket": "
 bH0q4n8XQOWbjStHsVCzAzS3fbkV2mbx-HUC8Bxw7r31HXcXRvPp4xWORxSo98N4
 M6uklYZEEC5OvlYBQ0kETabpBz7-dYo7nYCD6yCFlvE"}}]}}
```

Securing the Connection

- Current
  - Just relies on TLS transport
  - Requires a TLS stack for device binding
    - But not necessarily on the target device

- Future
  - Add ephemeral DH for additional security
  - Lightweight version for constrained devices

Why not DTLS

- Too much complexity
  - DTLS has all the features of TLS plus extras
  - There is no escape from the complexity
  - DNS should be a minimal service

- Missing required features
  - User registration process
  - Device binding mechanism.

UDP Request

```
struct {
    TransactionID       transactionID;
    SecurityContextID   securityContextID;
    opaque
encryptedPayload<1..65535>;
    opaque              authenticationCode<1..255>;
    } Request;
```

UDP Response

```
struct {
    TransactionID      transactionID;
    uint8              index;
    uint8              maxIndex;
    uint16             clearResponse;
    opaque             encryptedPayloadSegment<0..65535>;
    opaque             authenticationCode<1..255>;
    } Response;
```

Related Work

- SXS-Confirm
  - JSON based 2nd factor protocol
    - ‘New Printer X wants to join your network Accept/Reject’
- Omnibroker
  - JSON based Meta-discovery protocol
    - ‘Tell X how to connect to Y using protocol Z’
- Omnipublish
  - JSON based provisioning protocol
    - ‘I offer service P on address Q using credential R’
    - ‘Give me credential R to offer service P on address Q’