

*afnic*

# State of the "DNS privacy" project

*Stéphane Bortzmeyer*

AFNIC

*bortzmeyer@nic.fr*

*afnic*

# State of the "DNS privacy" project

*Stéphane Bortzmeyer*

AFNIC

*bortzmeyer@nic.fr*

*afnic*



# Warsaw OARC workshop

- May 2014: talk of the “DNS privacy project”
- See the slides for the context



# A brief reminder

- 1 A DNS query reveals what you're interested in  
(`_bittorrent-tracker._tcp.domain.example`)
- 2 Eve can be on the wire (sniffer) but also in the name servers  
(“DNSCrypt doesn't prevent third-party DNS resolvers from logging your activity”, to quote the DNSCrypt documentation)

# Encryption is not everything

- 1 Send as little data as possible (RFC 6973, section 6.1)
- 2 Encrypt it

1) is necessary against the evil name server. 2) is necessary against third-party sniffers.

# State of the project

On the standards side:

- 1 RFC 7626 “DNS Privacy Considerations” published
- 2 RFC 7816 “DNS Query Name Minimisation to Improve Privacy” published (status “experimental”)
- 3 Future RFC “Specification for DNS over TLS” approved by IESG, in the RFC Editor queue (status “standard”)
- 4 A few drafts are still under discussion

# Running code

Stolen from Sinodun <https://portal.sinodun.com/wiki/display/TDNS/DNS-over-TLS+implementations>

<i>Client/Server</i>	Client - Stub				Client - Recursive			Server - Recursive		Server - Auth	
<i>Software</i>	<i>ldns</i> (drill)	<i>digit</i>	<i>getdns</i>	BIND (dig)	<i>getdns*</i>	<i>Unbound</i>	BIND	<i>Unbound</i>	BIND	<i>NSD</i>	BIND
Port based TLS		✓	✓		✓	✓		✓			
TCP fast open**		✓	✓		P						
Connection reuse		✓	✓	✓	WIP	WIP		✓	✓	✓	✓
Pipelining***	n/a	✓	✓	n/a				✓	✓	✓	✓
OOOP***	n/a	✓	✓	n/a				WIP	✓		
TLS authentication			✓			2016		✓			
EDNS0 Padding			✓								
EDNS0 Keepalive			✓			2016					

# Minimising the QNAME

- ① No need to send the full QNAME to the authoritative name servers
- ② Ask NS fr to the root name servers instead of AAAA  
`www.internautique.fr`
- ③ In resolvers only (no change of the protocol)



# Implementation of QNAME minimisation

- Unbound (version  $\geq 1.5.7$ ). Off by default. See Ralph Dolman's talk.
- Knot Resolver (currently beta). On by default. See Ondřej Surý's talk.

# QNAME minimisation with Knot

dig -x of an IPv6 address, seen by tcpdump:

```
> 38773% [1au] NS? aRpA. (33)
> 22056% [1au] NS? Ip6.aRPa. (37)
> 43002% [1au] NS? 2.ip6.arPA. (39)
```

# The annoying broken name servers

Knot retries with full QNAME when receiving NXDOMAIN:

```
> 24014% [1au] A? WwW.UpENn.edU. (42)
< 24014*- 2/0/1 CNAME www.upenn.edu-dscg.edgesuite.net., RRSIG (270)
> 52576% [1au] NS? edGeSUItE.NEt. (42)
< 52576- 0/17/15 (1034)
> 22228 [1au] NS? EdU-DScG.EdGesUITe.nET. (51)
< 22228 NXDomain*- 0/1/1 (114)
> 1355 [1au] A? WWW.UPenN.edu-dSCG.EdgESuItE.net. (61)
```

# No way to know if it is an ENT

(ENT = Empty Non-Terminal domain name) Request for  
www.long.verylong.detail.example:

```
> 19881% [1au] NS? ExaMpLE. (36)
[NXDOMAIN received]
> 40708% [1au] AAAA? www.LONg.VeRyLONG.DEtaiL.eXamPLE. (61)
```

(Same thing with Unbound)

```
< 33070 NXDomain*- q: NS? example. 0/6/1
> 31355% [1au] A? www.long.verylong.detail.example. ar: . OPT UDPsize=4
```

# Encrypting data

- ① DNSCurve/DNSCrypt.
- ② TLS. Relies on the well-known TLS. Main version, above TCP and therefore persistent connections (RFC 7766). Port 853.



# DNSCrypt

`https://dnscrypt.org/`

- Not a standard (but there is running code, and deployment)
- Encrypt DNS requests to a trusted resolver
- Uses UDP
- No cryptographic agility
- Resolver authenticated by its public key (last column in the CSV file)
- Free software
- Many public resolvers (come and go quite often)

# DNScript encrypted

```
17:26:41.720678 IP (tos 0x0, ttl 64, id 59095, offset 0, flags [+], pro  
    192.168.2.9.33725 > 212.47.228.136.443: UDP, bad length 1664 > 1472  
17:26:41.721372 IP (tos 0x0, ttl 64, id 59095, offset 1480, flags [none]  
    192.168.2.9 > 212.47.228.136: ip-proto-17  
17:26:41.794366 IP (tos 0x0, ttl 64, id 59102, offset 0, flags [none],  
    192.168.2.9.33725 > 212.47.228.136.443: [bad udp cksum 0x8143 -> 0x  
17:26:41.840503 IP (tos 0x0, ttl 50, id 52891, offset 0, flags [none],  
    212.47.228.136.443 > 192.168.2.9.33725: [udp sum ok] UDP, length 56
```

# TLS with Unbound

Implemented for a long time (1.4.22?)

```
ssl-service-key: "/etc/unbound/privatekeyfile.key"  
ssl-service-pem: "/etc/unbound/publiccertfile.pem"  
interface: 2001:db8:1::dead:beef@853  
ssl-port: 853
```

If you don't know OpenSSL :

```
openssl req -x509 -newkey rsa:4096 \  
-keyout privatekeyfile.key -out publiccertfile.pem \  
-days 1000 -nodes
```



# Unbound starts and answers

```
unbound[12959:0] debug: setup TCP for SSL service
...
unbound[12959:0] debug: SSL DNS connection ip4 192.168.2.1 port 52185 (
...
unbound[12959:0] debug: Reading ssl tcp query of length 59
```

# And if I don't have a server?

`https://portal.sinodun.com/wiki/display/TDNS/  
DNS-over-TLS+test+servers`

Testing only, no production (one serves only one zone)

# First client, digit

<https://ant.isi.edu/software/tdns/index.html> Not fully maintained? (Strange errors, no IPv6)

```
% ./digit/digit -f domains-short -t tls -r 192.168.2.9 -p 853
#fsdb index t_complete t_avg t_individual t_sum t_mean id
query_send_ts response_receive_ts program_start_ts
1 0.614152 0.614152 0.614152 0.614152 0.614152 19383
1459097697.585573 1459097698.199725 1459097697.585572
```

## Second client, getdns

<https://getdnsapi.net/>, see Sara Dickinson's talk

```
% ./getdns/src/test/getdns\_query @192.168.2.9 -s -A -l L \  
    www.bortzmeyer.org
```

```
...
```

```
Response code was: GOOD. Status was: At least one response was returned
```

(-s: stub resolver, -A: ask for addresses, -l L: TLS transport)

# TLS in Go

<https://miek.nl/2014/August/16/go-dns-package/>

```
c := new(dns.Client)
c.Net = "tcp-tls"
if *insecure {
    c.TLSConfig = new(tls.Config)
    c.TLSConfig.InsecureSkipVerify = true
}
in, rtt, err := c.Exchange(m, net.JoinHostPort(ns, "853"))
```

# The pleasures of TLS authentication

- 1 No auth.: vulnerable to Mallory (the man in the middle)
- 2 Auth.: lots of trouble (“do you really trust this expired auto-signed certificate using SHA-1?”)
- 3 No hard rules: different profiles for authentication

```
% ./tls my-resolver internautique.fr  
Error in query: x509: certificate signed by unknown authority
```

```
% ./tls -k my-resolver internautique.fr  
(time 43051  $\mu$ s) 2 keys. TC=false
```

## See the traffic

```
% tshark -n -d tcp.port==853,ssl -r /tmp/dnstls.pcap
 4   0.002996 192.168.2.9 -> 192.168.2.9  SSL Client Hello
 6   0.594206 192.168.2.9 -> 192.168.2.9  TLSv1.2 Server Hello, Certif
 8   0.734094 192.168.2.9 -> 192.168.2.9  TLSv1.2 Client Key Exchange
16   0.751614 192.168.2.9 -> 192.168.2.9  TLSv1.2 Application Data
17   0.759223 192.168.2.9 -> 192.168.2.9  TLSv1.2 Application Data
```

(With Wireshark, Analyze → Decode as → SSL)

# (Provisional) Conclusion

- ① We have running code
- ② Deployment almost zero, currently





*Merci !*

*afnic*

[www.afnic.fr](http://www.afnic.fr)  
[contact@afnic.fr](mailto:contact@afnic.fr)

*afnic*