On Robust Covert Channels Inside DNS

Lucas Nussbaum, Pierre Neyron and Olivier Richard

Laboratoire d'Informatique de Grenoble / INRIA

Introduction

Many networks with restricted Internet access :

- Wireless access points in hotels and airports
- Censored Internet access in some countries

Question : How can one get full Internet access ?

Idea : Leverage one of the unfiltered protocols

DNS ?

DNS : perfect protocol ?

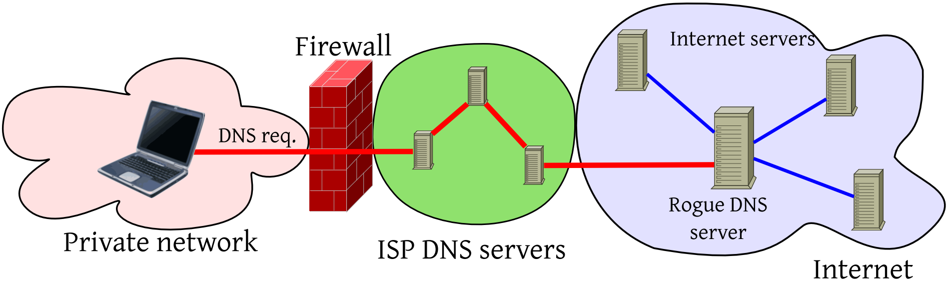
- (Almost) never filtered
    - And cannot reply with wrong results because of cache

But was not designed for tunnelling data

- Need to work around several DNS limitations

DNS covert channels : principle

- Hide data into DNS requests and replies
- Communicate with a rogue DNS server on the Internet



Existing implementations of DNS tunnelling

Not a new idea :

IP over DNS tunnels :

- NSTX
- Iodine

TCP over DNS tunnels :

- OzymanDNS
- dns2tcp

⇒ Compromises between protocol compliance and efficiency

DNS record types

Example :

```
> CNAME ? www.google.com.
< q : CNAME ? www.google.com.
    www.google.com. CNAME www.l.google.com.
```

Name being queried : only text (A-Z, a-z, 0-9, "-")

Record type being queried (implies type of reply) :

- A : only 4 bytes of data !
- CNAME : text with additional requirements (valid DNS name)
- TXT : any kind of data **[NSTX, OzymanDNS, dns2tcp]**
  - But not many real-life uses $\Rightarrow$ often blocked
- NULL : for experimental purposes **[Iodine]**
  - No known real-life usage

DNS extension : EDNS0

- Specified in RFC 2671
- Allow for larger packets
- Used by **Iodine** and **OzymanDNS**
- Not many real-life uses

  ⇒ can easily be blocked by ISPs

Data encoding in queries and replies

DNS names :

- A-Z, a-z, 0-9, "-" => 63 characters
- DNS servers "should" preserve case if possible

2 solutions :

- Base32 (need 32 characters)
  - Less efficient, but protocol compliant **[OzymanDNS]**
- Base64 (need 64 characters)
  - Adding another, invalid character :
    - "_" **[NSTX, Iodine]**
    - "/" **[dns2tcp]**
  - Using an escaping system
    - But packet length would vary

Evaluation of existing solutions

All solutions tested on several networks
(academic, home ISP, hotels, airports, etc...)

Each of them failed to work in some cases

⇒ Too many compromises with protocol compliance ?

⇒ Build our own solution ?

TUNS

IP over DNS tunnel

- Standard-compliance : uses CNAME records and Base32
- Handle poor network conditions :
  - Does not split IP packets
    - Lower MTU instead
  - Handle duplicate replies
  - Efficient polling mechanism

Example packets

Data packet from client to server :

```
dIUAAAVAAABAAAQABJ5K4BKBVAHAKQNICBAAAOS5TD4ASKPSQIJEM7VABAAEASC.
MRTGQ2TMNY0.domain.tld: type CNAME, class IN
```

The client sends a short query that the server will use to send a reply :

```
r882.domain.tld: type CNAME, class IN
```

The server acknowledges the data that was sent :

```
Queries
 dIUAAAVAAABAAAQABJ5K4BKBVAHAKQNICBAAAOS5TD4ASKPSQIJEM7VABAAEASC.
 MRTGQ2TMNY0.domain.tld: type CNAME, class IN
Answers
 dIUA[..]0.domain.tld: type CNAME, class IN, cname l4.domain.tld
```

The server sends a reply containing data to the client :

```
Queries
 r882.domain.tld: type CNAME, class IN
Answers
 r882.domain.tld: type CNAME, class IN, cname dIUAAAVCWIUAAAQABH
 VCY2DMO2HQ7EAQSEIZEEUTCOKBJFIVSYLJOF4YDC.MRTGQ2TMNY0.domain.tld
```

Efficiently polling the server

Problem :

- Server sends data to client using DNS replies
- To send a DNS reply, the server needs a query

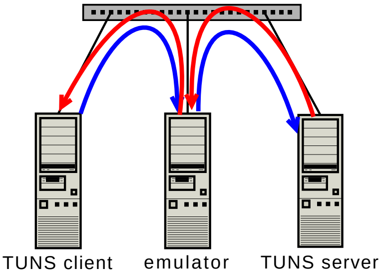
Solution : On regular intervals, send a DNS query to the server
The server answers with data or indicates that there's no data

Optimization : **[NSTX and TUNS, but not Iodine]**
If there's no data, wait for a while.
Data might arrive in the meantime.
From the client POV, the server simply looks busy.
$\Rightarrow$ Improves perceived latency significantly

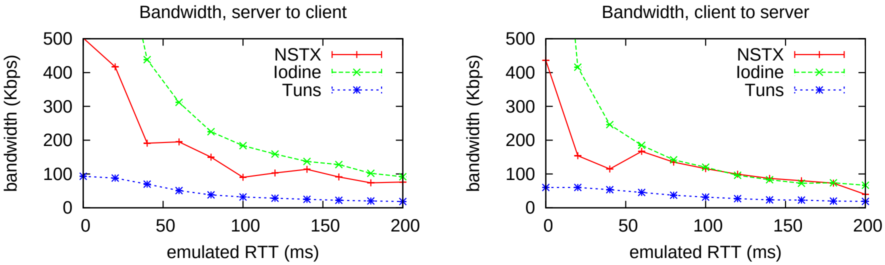
Performance evaluation

Compared NSTX, Iodine and TUNS using a network emulator

- Measured the tunnel's latency and bandwidth with varying network latency
- Also when facing degraded network conditions (5% packet loss, variable latency causing packet reordering)

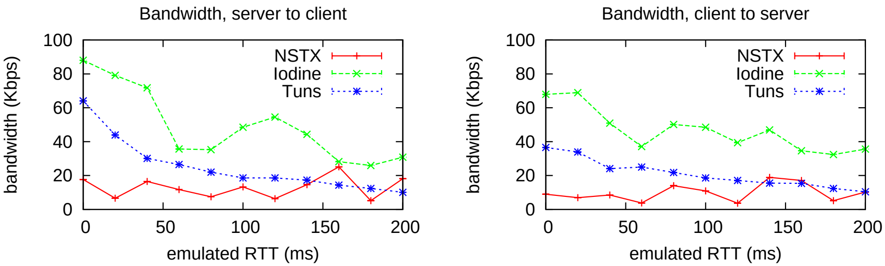


Results : bandwidth



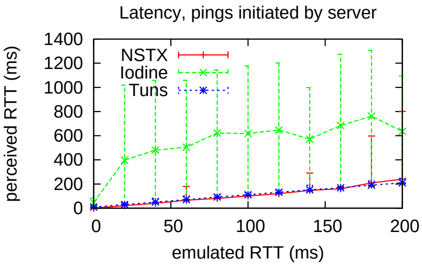
⇒ TUNS is slower than the other implementations

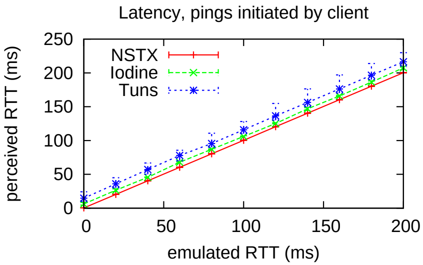
Results : bandwidth, with loss / reordering



... but stays more stable when network conditions are degrated, and outperforms NSTX

Results : latency





⇒ Iodine’s polling mechanism is inefficient

Conclusion

Exposed the various challenges faced by DNS covert channels

Described TUNS, our IP over DNS tunnel

- Slower that the other implementations in some cases
- But uses only standard DNS features
  - Harder to block by system administrators
    - Remaining solution : traffic shaping
  - Worked on all networks we could try
    - Except those with broken DNS, of course

Future Work

- Tuning of tunnel parameters from the client-side
  - Packet length, DNS record type, encoding
- Automatic detection of best parameters
- Headers compression $\Rightarrow$ more space for data
- TCP tuning to better handle packet re-ordering
  - Very frequent over DNS
- Encryption of data being transmitted