DNS Tunneling

Ken Bodzak

CS615ASA - Spring 2013
Why tunnel?

Tunneling Protocol: ... When one network protocol encapsulates a different protocol.  
[From: en.wikipedia.org/wiki/Tunneling_protocol]

We can run protocols over an 'incompatible' network.

We can use an unprotected protocol inside of a secure channel. (Think SSH)
If we have access to DNS, we can run another protocol inside of it.

We can send data out of our network using DNS requests.

We can receive data into our network inside of DNS responses.
How does it work?

When we request a domain to be resolved, the server responsible for knowing the 'answer' should eventually be contacted.

This allows us some control over where our DNS request gets sent.

If the server responsible for servicing a requested domain is under our control, we can use it as our tunnel endpoint.
How does it work?

Our 'tunneling' server can then respond with a DNS response.  
(That should eventually reach the client)

This allows us to utilize DNS to establish two-way communication with our external server.
What do we need?

**Client Software**
- Access to DNS from the network
- Write 'tunneled' data into DNS requests
- Read 'tunneled' data from DNS responses

**Server** (Fake DNS server / tunnel endpoint.)
- Responsible for resolving our target domain.
- Read/Write 'tunneled' data
Requests

We can hide our messages in the actual domain being requested.

arbitrarydata.tobesent.tunnel.myhost.net

When the server responsible for tunnel.myhost.net receives this request it will know to interpret:

arbitrarydata.tobesent

as a message and not a subdomain.
Responses

The server can now send a DNS response that includes a response to the message.

We can use different record types to make this process easier.

*ie.*

A Record -> Returns a 32b IP address

TXT Record -> Returns arbitrary ASCII
Example

Request: myquestion.tunnel.mydomain.net

TXT response: myanswer

Asks Question

Gets Answer
Advances

Utilize Different record types (NULL records)

Different encodings supported by certain DNS software. (Allows for more throughput)

Abuse slack-space rather than using subdomains (PSUDP)[1]
Why do we care?

- Data exfiltration
  * Corporate espionage
  * Malware (C2 Traffic)

- Circumventing network policies and rules

- Evading captive portals
DNS Tunneling provides a communication channel in and out of your network.
- Can we detect it? monitor it? prevent it?

One of the more popular use cases:
- Allowing an unauthenticated host to use restricted services.
- ie. Getting around captive portals
Captive Portal  (*stevens guest*)

May block most services, but often not DNS!

What can we do about it?

Most Important: Control your DNS traffic
  - Network Setup

Analyze your DNS traffic
  - Detection

Traffic shaping / Shunning
  - Respond
Controlling DNS Traffic

Ensure that you know what DNS traffic is coming into and out of your network.

- Internal DNS server / proxy
- Split DNS
  (Different DNS for different hosts.)
  Different answers to the same query.

This does **NOT** prevent all DNS tunneling, but will help us to manage it.
Detection

- Lots of DNS traffic, likely from a single host.

- Arbitrary subdomains being requested
  * Character frequency analysis [2]

- Odd record types being used. (NULL, TXT)
  * Restrict the use of these records

  (could break other things!)
Ex. Traffic per host

DNS Traffic

Host

A B C D E

Probably up to something...
Ex. Character Frequency

[2] Born, Gustafson
Countermeasures

- Un-authenticated hosts have restricted DNS (via split DNS)

- Traffic shaping / Shunning
  'Restrict' hosts who abuse DNS traffic
  Deny certain record types (NULL, TXT)

- Analyze and block suspicious DNS traffic (Anomaly detection)
Downsides

DNS is very helpful to end-users and lots of applications utilize it.

- Restricting DNS can cause collateral damage

Anomaly detection and traffic shaping is prone to false positives

- Wrongfully deny service to your users
Final Thoughts

DNS Tunneling is a nifty tool, but also a real problem in network security.

The usefulness and widespread deployment of DNS make it hard to modify, disable, or restrict on a large scale.
- This places the burden of coping with its 'vulnerabilities' on administrators.
References
