Black Ops of DNS
Chaos Communications Camp 2003

Dan Kaminsky, CISSP

Senior Security Consultant

IP Telephony | Contact Centers | Unified Communication | Services

Your Friendly Investigator

- **Who am I?**
  - **Senior Security Consultant, Avaya Enterprise Security Practice**
  - **Author of “Paketto Keiretsu”, a collection of advanced TCP/IP manipulation tools**
  - **Speaker at Black Hat Briefings**
    - **Black Ops of TCP/IP series**
    - **Gateway Cryptography w/ OpenSSH**
  - **Protocol Geek**

The Subject Of Our Investigation

- DNS: The Domain Name System
  - Created by Paul Mockapetris in 1983
  - Fast, easy, accurate way to, given a host's name, find out it's number
    - The Internet doesn't run on names any more than the telephone network does – everything is numbered for efficiency.
  - "Internet's equivalent of 411"
    - Actually more critical – people will keep the same phone number for years, while some services change their IP addresses constantly
      - Was becoming a management nightmare to pass around lists of names/numbers
      - "Call 411 by default"
    - Internet has no secure mechanism for sending a fail/redirect message ("the number you've called has been disconnected, the new number is…")

The Nature Of Our Investigation

- The Question: Is it possible for DNS to do anything more interesting than return numbers from names?
  - Simple answer: Of course, it can return mail servers, names from numbers, SPF records, etc.
  - Better answer: Why do you ask?
    - Second oldest "uncontested protocol" for what it does
      - Telnet's moved to SSH, Gopher and FTP moved to HTTP
      - Only SMTP is in a similar class
    - Globally deployed, universally employed
    - Routes everywhere, through pretty much any network.
    - *Was heavily queried during recent MS Blaster worm.*
  - Ultimate answer: Yes, or this would be a very short talk!
- The strategy: Does DNS have any unexpected similarities – "*homologies*" – to other protocols I consider interesting?
  - Since I consider network tunneling interesting…

The Totally Expected And Well Known Homology Within The Structure of DNS

- DNS servers can carry arbitrary data
  - Not just English host names and IP addresses!
  - Many constraints are enforced by the DNS architecture
    - Total packet size must remain less than 512bytes (by default)
    - Downstream Channel: Content of arbitrary TXT record
      - Very flexible near-arbitrary content field, supports multiple independent order-maintained subrecords <256bytes long
      - Data *probably* needs to be text compatible, so we Base64 this data
      - Could use any of a number of other channels – MX records worked very well too
    - Upstream Channel: Name of host being looked up
      - <253 characters, <64 characters between dots, only 63 valid characters
      - Bits are bits, so we can use Base32 to put ~110 arbitrary bytes in a query name
  - I'm not the first person to put odd things in DNS; this has been done for everything from Bittorrent Seed Distribution to Online Calculators

Arbitrary Data Detail

- What arbitrary encapsulation looks like
  - Base64 in TXT Records (~220 bytes per packet)
    - dig 0.b64.alt.doxpara.com txt

      …

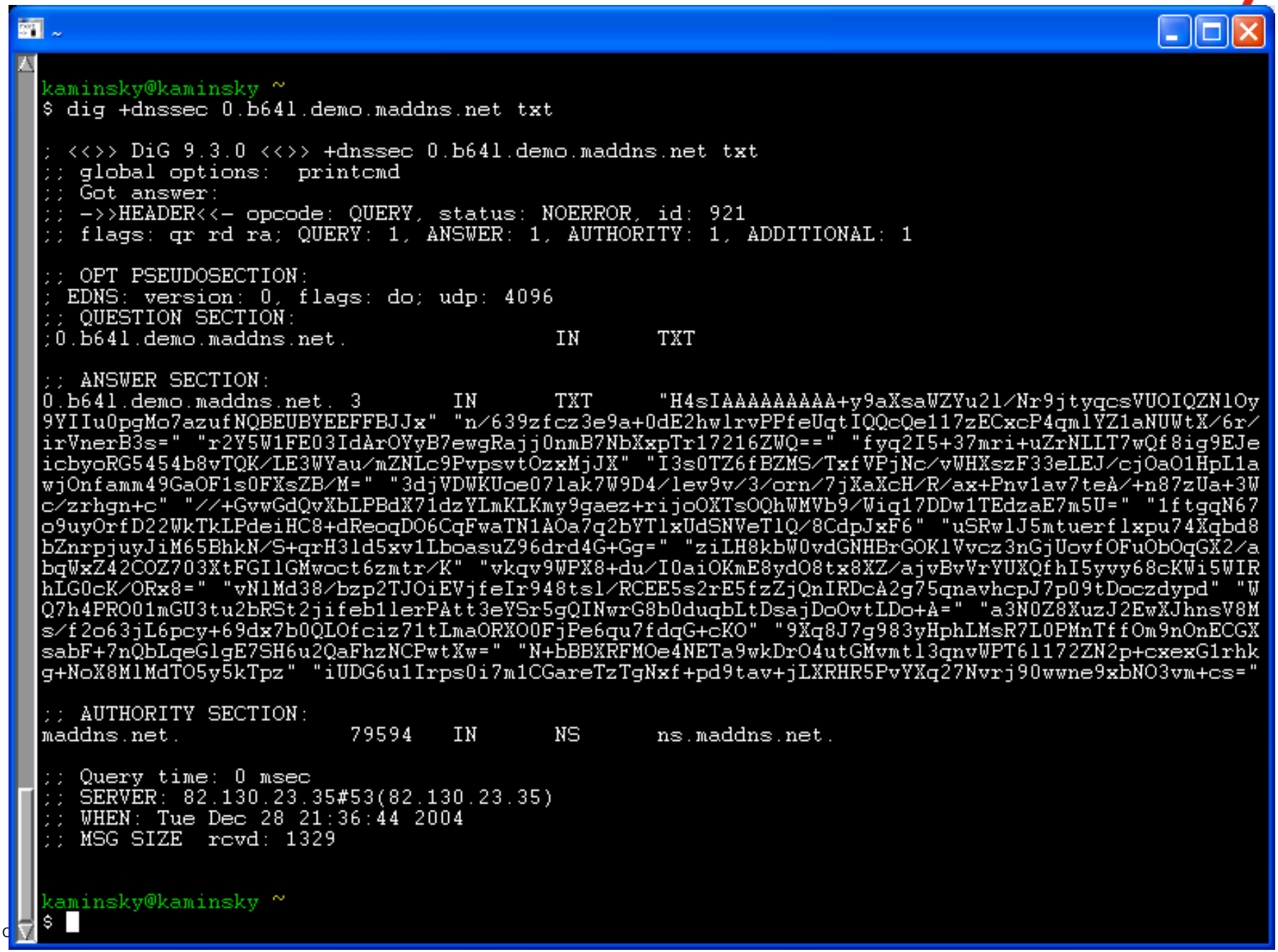
      0.b64.alt.doxpara.com. 60 IN TXT "lyEvdXNyL2Jpbi9wZXJsIAoKJFZFUlNJT04gPSAwLjE7Cgp1c2UgRmNudGw7CnVzZSBOZXQ6OkRO\010UzsKdXNlIE5ldDo6RE5TOjpOYW1lc2VydmVyOwojdXNlIExXUDo6U2ltcGxlOwp1c2UgTFdQOjpV\010"
  - Base32 in CNAME Records (~110 bytes per packet)
    - dig 0.get.b32.alt.doxpara.com

      …

      0.get.b32.alt.doxpara.com. 0 IN CNAME orugs4zanfzsayjaorsxg5ban5tca5dimuqgk3lfojtwk3tdpeqge4tpmfsggyl.toqqhg6ltorsw2cq.0.set.b32.alt.doxpara.com.
    - orugs4zanfzsayjaorsxg5ban5tca5dimuqgk3lfojtwk3tdpeqge4tpmfsggyl.toqqhg6ltorsw2cq.0.set.b32.alt.doxpara.com. 0 IN A 1.0.0.0
      - DNS Compression – no redundant data

Under Development – Hi Bandwidth Payloads

- 512 byte limit exists – but can be waived
  - EDNS0 OPTion for DNSSEC
  - Can specify >MTU limits (IP Fragmentation?)
  - Able to go from 220 bytes down to 1024 bytes – near 5x improvement!


~
kaminsky@kaminsky ~
$ dig +dnssec 0.b64l.demo.maddns.net txt
; <<>> DiG 9.3.0 <<>> +dnssec 0.b64l.demo.maddns.net txt
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 921
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;0.b64l.demo.maddns.net. IN TXT
;; ANSWER SECTION:
0.b64l.demo.maddns.net. 3 IN TXT "H4sIAAAAAAAAA+y9aXsaWZYu2l/Nr9jtyqcsVUOIQZN1Oy
9YIIu0pgMo7azufNQBEUBYEEFFBJJx" "n/639zfcz3e9a+0dE2hwlrvPPfeUqtIQQcQe117zECxcP4qmlYZ1aNUWtX/6r/
irVnerB3s=" "r2Y5W1FE03IdArOYyB7ewgRajj0nmB7NbXxpTr17216ZWQ==" "fyq2I5+37mri+uZrNLLT7wQf8ig9EJe
icbyoRG5454b8vTQK/LE3WYau/mZNLc9PvpsvtOzxMjJX" "I3s0TZ6fBZMS/TxfVPjNc/vWHXszF33eLEJ/cjOaO1HpL1a
wjOnfamm49GaOF1s0FXsZB/M=" "3djVDWKUoe07lak7W9D4/lev9v/3/orn/7jXaXcH/R/ax+Pnv1av7teA/+n87zUa+3W
c/zrhgn+c" "//+GvwGdQvXbLPBdX71dzYLmKLKmy9gaez+rijoOXTsOQhWMVb9/Wiq17DDw1TEdzaE7m5U=" "1ftgqN67
o9uyOrfD22WkTkLPdeiHC8+dReoqDO6CqFwaTN1AOa7q2bYT1xUdSNVeT1Q/8CdpJxF6" "uSRw1J5mtuerf1xpu74Xqbd8
bZnrpjuyJiM65BhkN/S+qrH31d5xv1LboasuZ96drd4G+Gg=" "ziLH8kbW0vdGNHBrGOK1Vvcz3nGjUovfOFuObOqGX2/a
bqWxZ42COZ703XtFGI1GMwoct6zmtr/K" "vkqv9WPX8+du/I0aiOKmE8ydO8tx8XZ/ajvBvVrYUXQfhI5yvy68cKWi5WIR
hLG0cK/ORx8=" "vN1Md38/bzp2TJOiEVjfeIr948tsl/RCEE5s2rE5fzZjQnIRDcA2g75qnavhcpJ7p09tDoczdypd" "W
Q7h4PRO01mGU3tu2bRSt2jifeb1lerPAtt3eYSr5gQINwrG8b0duqbLtDsajDoOvtLDo+A=" "a3NOZ8XuzJ2EwXJhnsV8M
s/f2o63jL6pcy+69dx7b0QLOfciz71tLmaORXO0FjPe6qu7fdqG+cKO" "9Xq8J7g983yHphLMsR7L0PMnTffOm9nOnECGX
sabF+7nQbLqeG1gE7SH6u2QaFhzNCPwtXw=" "N+bBBXRFMOe4NETa9wkDrO4utGMvmt13qnvWPT61172ZN2p+cxexG1rhk
g+NoX8M1MdTO5y5kTpz" "iUDG6u1Irps0i7m1CGareTzTgNxf+pd9tav+jLXRHR5PvYXq27Nvrj90wwne9xbNO3vm+cs="
;; AUTHORITY SECTION:
maddns.net. 79594 IN NS ns.maddns.net.
;; Query time: 0 msec
;; SERVER: 82.130.23.35#53(82.130.23.35)
;; WHEN: Tue Dec 28 21:36:44 2004
;; MSG SIZE rcvd: 1329
kaminsky@kaminsky ~
$

Unexpected Homologies Within The Structure of DNS

- DNS Proxies
  - Those that do not know will ask those that do
  - “Recursive Lookup”
  - Lookups are heavy processes; was necessary to centralize the work
- DNS Caches
  - If one name server does proxy for another, results are not ephemeral, rather they’re cached for a definable amount of time (up to a week in most implementations)
- DNS Routes
  - Those that don’t know, and don’t want to ask those that do, can instead reply with a route recommendation of who else to speak to. Those that receive route recommendations will generally follow them.
  - Used to implement the DNS hierarchy
    - .com routes to doxpara.com routes to www.doxpara.com
  - “Iterative Lookup”

Pause for Impact

- So, in summary:
  - We can move arbitrary data.
  - We can bounce it off arbitrary servers.
    - Over 150K on 64.*, over 2M total
  - We can route it in arbitrary ways
  - We can store data in a hop-by-hop basis
- "DNS is a globally deployed, routing, caching overlay network running astride the entire public *and private* Internet."
  - Yeah, that's pretty interesting
  - Terms usually used to describe peer to peer networks

Weapon of Choice

- Under Development: OzymanDNS
  - Toolkit for advanced DNS investigation
  - Built using Perl's Net::DNS library
    - Shockingly mature packet mangling library for DNS manipulation
  - Implements many of the concepts we're discussing here today
  - Server: Nomde – Experimental DNS Server
  - Clients: You'll see

DNS Tunneling[0]: Starting Simple

- Is it possible to achieve useful network connectivity out from even extraordinarily egress-filtered networks?
  - Answer: Yes – by encapsulating all traffic inside of necessarily passed DNS queries
  - Constraint: Must remain 100% RFC Compliant
- What makes this particularly interesting?
  - "Routes everywhere, through pretty much any network."
    - Escapes most forms of egress restriction
      - And more.
  - DNS was not designed to do this, so this makes a particularly broken network to design connectivity solutions for
  - Good introduction; useful

DNS Tunneling[1]: Who's doing it?

- **Who?**
  - **NSTX most popular**
    - **Creates a "virtual network device" that routes IP (actually, Ethernet frames) over DNS**
    - ***Linux Only***
  - **Rumors of various botnets / malware using DNS as a covert channel**

DNS Tunneling[2]: Another Approach

- Starting “Simple”
  - NSTX requires kernel cooperation to get at IP
  - Lets make something that doesn’t require the kernel, but still allows remote networking
    - Remote Networking: “I’m on this network, but all my traffic is routed through that network over there – preferably securely”
    - Normally done with VPNs (also kernel level)
    - SSH Dynamic Forwarding allows secure remote networking over a single TCP port (Poor Man’s VPN)
    - So lets start with SSH over DNS

DNS Tunneling[3]: Problems

- DNS is not TCP
  - TCP moves bytestreams, DNS moves records
    - Blocks of data
  - TCP lets either side speak first, while in DNS, the server can only talk if the client asks something
  - TCP is 8 bit clean, while DNS can only move a limited set of characters in each direction (Base64 / Base32)
  - This seems so familiar…

DNS Tunneling[4]: Mini-HTTP

- The semantics of DNS are surprisingly similar to those of HTTP
  - The BIG difference: Absurdly small packet sizes
    - Almost but not quite powers of 2
    - grr
- Many tools have been written with the “lets tunnel everything over HTTP” methodology because it gets through firewalls easier (see first point)
  - SOAP (RPC over HTTP)
  - GNU httptunnel (TCP Stream over HTTP)
- Since DNS has similar semantics, we can pull off similar feats
  - droute: DNS Stream Router

DNS Tunneling[5]: Routing through droute

- Different directions have different semantics
  - Upstream: Generally limited bandwidth (both needed and available), can send data on demand
    - Standard lookups of long Base32 encoded names
  - Downstream: More bandwidth, but must poll for new content
    - TXT lookups returning Base64'd data
      - Address contains byte offset from beginning of stream
      - Server manages session state
    - If a poll fails to return any data, triple the delay until the next poll (up to a maximum)
    - Set delay to minimum whenever data is sent
- Present implementation is "single-threaded" – only one packet can be outstanding at any given time
  - This is a temporary implementation bug.
    - Present high-speed status: 16K/s throughput, 20% packet loss
    - Every packet bouncing off a different public nameserver

So Egress Is Feasible…

- **Is it possible to achieve useful network connectivity into even extraordinarily ingress-filtered networks?**
  - **That's a much more complicated answer**
    - **But no is just two letters…**

Alice Wears Two Hats.

- Alice-as-requestor: “What’s the number for foo.com?”
- Alice-as-responder: “Oh, you don’t know? It’s 1.2.3.4.”
  - When does Alice become the responder? When the DNS hierarchy routes a name lookup request to her.
    - .com: foo.com is hosted by alice
    - Alice: foo.com is 1.2.3.4
  - Alice as responder can manipulate routes and caches
- 16-32 levels of redirection are allowed
  - Once someone’s been forwarded to Alice, they can be forwarded again…to any IP address.

DNS Source Routing[0]: The Problem At Hand

- DNS trusts the hierarchy to tell it where to route to next
  - The hierarchy is public
  - What the DNS server can route to isn't necessarily
    - A recursive, dual homed DNS server can be made to route requests back into an internal private network
    - Only DNS/53 traffic can be so proxied
      - Custom internal malware – no more polling, it now accepts incoming connections
        - » SSH over DNS, everything else here
      - Potential for attacks against internal active directory – attack must be contained to query name

DNS Source Routing[1]: HOWTO

- Nomde implements an echo service – requests to it are addressed according to the name being sought
  - This, of course, is very different than the basic concept of a directory – something that's written few, but read many.
- What it looks like
  - $dig 10-0-1-55.echo.alt.doxpara.com ns

    …

    10-0-1-55.echo.alt.doxpara.com. 15 IN NS ns.10-0-1-55.echo.alt.doxpara.com.
  - $dig ns.10-0-1-55.echo.alt.doxpara.com a

    …

    ns.10-0-1-55.echo.alt.doxpara.com. 11 IN A 10.0.1.55
- What it does
  - All requests to *.10-0-1-55.echo.alt.doxpara.com will be routed to 10.0.1.55, and all replies from the internal address will route back

DNS Source Routing[2]: The Easy Fix?

- "Fine, we'll just disable recursion on external interfaces"
  - "If she can't be Alice-the-Requester, she won't be Alice-the-Responder"
  - Responder-level access is required to issue the internal redirects
  - If we disable "external recursion" (i.e. the willingness for the DNS server to accept lookup requests from arbitrary hosts), the name server will never issue lookups to Alice, who will thus never be able to provide those evil redirects
- Question: Is it still possible to trigger lookups against an attacker-controlled nameserver, even from a server that's not accepting lookup requests from the public?
  - Answer: Yes, if you can manipulate trusted applications into starting a lookup chain that leads to you

DNS Source Routing[3]: Induced Recursion

- Eliminating open recursion (proxy for everyone) only eliminates one path towards the hierarchy being parsed
- Many open applications respond to requests with DNS lookups
  - Mail servers do MX lookups
  - *Everything* does PTR lookups (IP->Name)
    - SSHD
    - Apache
    - Probably Your IDS
      - How do I know? Scan 16M IP addresses, you'd be amazed how many hosts automatically reverse lookup your IP
      - It helps that I returned TTL=0 for my PTR record – system has to lookup for every packet that worries it ☺
      - Oh: Forensics people, I see your nslookups. Hi.
  - Just because a lookup *starts out* as a PTR requests doesn't mean it'll always remain one
    - Can route IP lookups to foreign name servers, which then need to be independently looked up.

DNS Source Routing[4]: Recursion-As-Route Intro

- ***This is the next generation attack still under development – details may change***
- **Once an application triggers the recursion engine on the DNS, the hierarchy is followed to the attacker**
- **The attacker does not answer the query directly, rather, forwards to an out-of-bailiwick name it also controls**
  - **Why? Because now the recursion engine must start from scratch, looking for a new name.**
  - **A new name like, "hello-ssh-are-you-there.attacker-new.com"**
  - ***This* name is NS-forwarded back inside the target network**
  - **The payload is contained in the query to the internal IP**
- **But how to respond?**

DNS Source Routing[5]: Recursion-As-Route Continues

- If it worked once, why not again?
- We forward the request to somewhere in the original attacker domain, which is now once again out of bailiwick
  - SSH-1-99-OpenSSH-3-7-1p1.attacker-old.com
- The name server has already cached the values for com and attacker-old.com, so it's quite quick to "lookup" the SSH initialization string ☺
- We can ping back and forth, repeatedly, until we hit the nameserver's limit for recursion depth for the original reverse name lookup
  - Best to eventually respond – with TTL=0 – so that another packet will start another lookup chain
  - We can of course respond with anything.

Aside: Don’t just log the PTR name, please.

- **nslookup 64.81.64.164 4.2.2.1**
  164.64.81.64.in-addr.arpa name = 127.0.0.1.
- **This does very bad things to your logs.**

More Risks of Open Recursion

- Widespread recursion actually isn't bad – it contributes to the stability of DNS, which by all rights should have collapsed ages ago – but maybe there's too much of a good thing
- Tunneling challenge: Is it possible for two people to communicate through DNS, without either of them controlling a server?
  - Can't upload records
  - Can cause recursive queries to occur
    - Recursive queries – that are successful – are cached
      - State change in the server
    - Recursion can be disabled by the requestor – "if you've already got the data, tell me, but if not, don't go out and look it up"
      - Externally visible state change in the server

Single-Bit Data Transfer[0]: HOWTO

- Sending:
  - Step 1: Split message into individual bits.
  - Step 2: For each byte that will be available for reading, do a recursive lookup against a “start bit” address.
  - Step 3: For each bit that is 1, do a recursive lookup against a wildcard-hosted name that identifies that bit.
- Receiving:
  - Step 1: Do a lookup for the first byte’s start bit. If set to 1…
  - Step 2: Do non-recursive lookups against names that map against all eight bits. Those names that return answers are 1, those that don’t are 0.
  - Step 3: Integrate bits into a byte and save. Increment byte counter and return to step 1.
- Deleting:
  - Optional: Simply do a recursive lookup to clear the 0’s

DNS Cache Snooping

- We are not the only users of the DNS cache
  - Other hosts use DNS to do useful things
  - We can thus track them being useful
  - Best paper on subject: Luis Grangeia “Snooping the Cache for Fun and Profit”
- Miname – DNS scanner – Able to DNS cache snoop on Internet Scale (find how many sites are looking up
  - Can detect not only that a service was used, but when

Single-Bit Data Transfer[1]: Implications

- **Bidirectionality**
  - **Simply use two separate channels, or timeshare one channel**
  - **It's just a shared medium**
- **Adaptability**
  - ***Anything* that can be changed by one and seen by another can be used to send data**
    - **Web Counters**
    - **IP ID counters in IP Stacks**
    - **Paul Revere's lamps in the tower – the British are Coming!**
  - **"Did it increment or not? Were there sharp spikes between 6:01 and 6:02?"**
  - ***"You can always send a bit"* – can we send more?**

Last Modified over DNS[0]

- Slashdot Story: "RSS IS OVERLOADING THE INTERNET!"
  - Lots and lots of news readers do "big heavy" HTTP queries every hour on the hour
  - Looks like a DDOS
  - They're interested in data only if it's new – check Last-Modified: value (or use If-Modified-Since:)
- Question: Is it possible to use DNS to solve this problem?
  - Could just put Last-Modified in DNS TXT
    - But it's just a time…and times are 32 bit ints…
    - You know what else is 32 bits? That DNS is really good at transmitting?

Last Modified over DNS[1]

- Representing modification dates as miscasted IP addresses
  - Set IP address to little endian representation of time since ~1973.
  - WARNING: WILL FAIL IN 2038
  - Extremely simple client code
- glance.c:
  - host = gethostbyname(argv[1]);
    time = (void *)host->h_addr;
- $ ./glance.exe www_cnn_com.glance.alt.doxpara.com
  1091117481 (Thu Jul 29 16:11:21 2004 GMT)
  - Caveat: Only support '.' (as '_') and '/' (as '-') in URL string
  - More requires url encoding or Base32

Domaincast[0]

- Question: Is it possible to distribute large files, quickly, using nothing but DNS?
  - Answer: Yes, by leveraging the sheer enormity of the DNS infrastructure.
- The basic concept
  - Normal DNS operation: Talk to your own server; it retrieves data on demand from the official server upstream. The data is consistent for everyone.
  - Domaincast DNS operation: Acquire a list of name servers from the data source. Each server has been seeded with n blocks of data. Acquire each server's cached subset of the entire file. Reassemble.
    - *This is possible. This is not necessarily a good idea. Don't try this at home.*

Domaincast[1]

- **Sidestepping Limited Resources**
  - **Capacity**
    - **20K/server = ~80 records @ 256bytes**
    - **700MB = Knoppix, a full Linux distribution**
    - **700MB / 20K = 35,000 servers required**
      - **Number of DNS servers detected in a single class A: +140,000 (More on this later)**
  - **Speed**
    - **1Kb/s * 35,000 = 35MB/s**
      - **Would require upload of ~3.5MB/s**
      - ~50% packet / formatting overhead = 17MB/s
    - **Not going to achieve peak bandwidth (it'd shred reliability) – but not going to get 1k/s either**
  - **Reliability**
    - **Caches are not determinate – they can drop on demand**
    - **So we use an M-of-N splitting algorithm compat. with small blocks**

Domaincast[2]

- Precached index of hops to be scanned is certainly the easiest mechanism for data distribution
  - 35,000 * 4 = 140K
  - 140K at 20K/s is 7 servers – can just distribute the hoplist over Domaincast too.
- There are more complex algorithms (in the earlier version of these slides) but they're ultimately not necessary ☺

But What Of Streams?

- Precached hop-lists can't really adapt to streaming data
- Question: Is is possible to transmit a live stream of data via the DNS infrastructure?
  - Answer: Of course – just populate on demand and issue pointers to the latest data. Do it right and you'll even cache correctly…

KDNS[0]

- Voice over DNS – TXT w/ Streaming Audio
- Speex codec supports Voice compression at ~2kbps (best *public* codec)
  - Ends up being about 276 bytes/second
    - 353 bytes w/ Ogg headers.
  - We can traffic 276 bytes per second through even extremely slow DNS servers
- Power to the Caching People
  - Use a TTL of WINDOW seconds (~20s)
  - All listeners behind the same DNS server will split the same stream

KDNS[1]

- Server HOWTO
  - <timestamp>.server.com
    - TTL=WINDOW
    - TXT (or MX) = 0.88s of audio
  - latest.server.com
    - CNAME to <window>.<timestamp>.server.com
    - TTL=0
    - This may be broken by resolvers with minimum TTL requirements (implemented to fight Dynamic DNS server loads)

KDNS[2]

- Client HOWTO
  - Retrieve latest.server.com, learn <window>.<timestamp>.server.com
  - Retrieve audio samples from <timestamp-window>.server.com up through <timestamp>.server.com
    - Prefer to start playing at a packet that has spent some time decrementing in the cache – binary search for oldest sample within the window
    - Add random jitter for the start sample – this way, everyone's at a different point in the stream, and if the "lead looker" drops, someone else will be next up to be first in line
- "geta" implementation doesn't support windowing or random jitter yet, but seems to work quite well regardless
  - Still caches!

KDNS[3]

- Alternate Implementation: Proxy-Friendly HTTP Streaming
  - Abandon DNS entirely; simply distribute 3-10s chunks of audio over HTTP and let proxy servers share them out
    - Can potentially use .PLS/.M3U chaining
  - Requires client cooperation, and greatly increases latency, but solves the (bandwidth) problem that proxies don't cache streams
  - Proxies support TTL's through the Expires: option
  - HTTP of course supports much higher bandwidth

Conclusions

- 1. DNS is globally deployed – you use it as a client, you probably depend on it as a server
- 2. As the rest of IP networking has become progressively more and more filtered, DNS's level of interconnectivity has (*for important functionality reasons*) remained constant, and in some ways outstrips the services offered by a completely unfirewalled host.
- 3. This connectivity can be used to offer a range of services, from encrypted VPN-style linkage, to a completely silent but remotely addressable trojan horse, to an unexpectedly useful distributed caching audio system.
- 4. DNS should not be disabled, re-engineered, blocked, or heavily interfered with at this time – but perhaps we can start paying closer attention to its traffic.