Implications of Netalyzr’s DNS Measurements

Nicholas Weaver (ICSI),
Christian Kreibich (ICSI),
Boris Nechaev (HIIT/TKK),
and Vern Paxson (ICSI & UC Berkeley)
Acknowledgements

“Where do I donate” - User Feedback

- This work sponsored by the National Science Foundation
  - With additional support from Comcast and Google
  - All opinions are those of the presenter, not those of the sponsors

- Netalyzr’s data set has multiple biases
  - A strong “geek” bias: Netalyzr’s users are often considerably more technical savvy than the normal population
  - A tendency towards flash-crowds: the context of the flash crowd affects the usage population
  - Participation in two separate IPv6 deployment tests
  - Now a common debugging tool for League Of Legends, an online game:
    - “None of that worked, my internet is teh pwned”...
      Run Netalyzr and post the results in the support forum
Key Insights Behind Netalyzr

- Java applets can perform a lot of activity by default:
  - Can use arbitrary TCP and (usually) UDP connections to the server hosting the applet
  - Can lookup arbitrary DNS A (address) records, but the result can only resolve to the hosting server’s IP or generate a security exception
- Java applets can do even more when “trusted” (the signature is accepted by the user):
  - Bypasses same origin for both DNS and connectivity
- Javascript can do other things:
  - Load third party images and validate success
  - Examine DOM to see if its in an iframe
- And our servers can do whatever it wants:
  - Any services, including deliberate protocol violations
  - Raw packet examination
Netalyzr’s Architecture:
“If I knew what all this meant, I’d be dangerous” -User Feedback

DNS and Netalyzr

Front End Server (@ ICSI)
- HTTP Server
- DNS Authority
- Storage
- ISP’s DNS Resolver

Server Pool (EC2)
- HTTP Server
- DNS Server
- Echo Servers
- Latency Server
- Local Storage
- MTU Server

ISP’s DNS Resolver

NAT? Something Else?

DNS Wildcard Server?

Summary of Noteworthy Events –
- Minor Aberrations
  - Certain TCP protocols are blocked on outbound traffic.
  - We received unexpected and possibly dangerous results when looking up important names.

Address-based Tests –
- NAT detection: No NAT Detected
- DNS-based host information: OK

Reachability Tests –
- TCP connectivity: Note
  - Direct TCP access to remote FTP servers (port 21) is allowed.
  - Direct TCP access to remote SSH servers (port 22) is allowed.
  - Direct TCP access to remote SMTP servers (port 25) is allowed.
Many tests focus on DNS operation:
  - Huge determinant of user experience

Initial tests:
  - Server properties through InetAddress.getByName(): Returned A records indicate properties
  - Important name lookup: 70+ names looked up with reverse validated on the server
  - Direct network probing for mandatory DNS proxies/firewalls

Later tests:
  - Additional resolver properties
  - In-network support for different DNS types
    - Queries directly to our server as a DNS authority
  - Probing of the NAT’s DNS gateway for supported DNS types
    - Now also probes the listed DNS resolvers
    - The probing includes more types, larger responses, large responses with EDNS, and type=CHAOS queries
Resolver Transport Issues:

- **EDNS0 MTU:**
  - If a resolver claims an MTU of >1800B:
    - Only 87% could actually process an ~1800B response
    - 98.5% could process a ~1300B response
    - The major problem is fragmentation, not “DNS==512B” assumptions
    - Resolvers need to detect and respond accordingly

- **TCP failover:**
  - 98% succeed in failing over to TCP
  - But .35% of the sessions *ignored* the truncate bit!?!?
    - The truncated reply includes a different A record in the UDP reply than the TCP reply
  - .51% had both a failure of TCP failover *and* a failure to handle fragments
  - QED: Keeping replies to <1500B is useful, but failures won’t be widespread if large replies (such as DNSSEC-authenticated certificates) become common

- **IPv6-only glue records:**
  - 5.1% of sessions were able to follow a AAAA glue record
**NXDOMAIN wildcarding**

- NXDOMAIN wildcarding is now **endemic**:
  - 27% of sessions show NXDOMAIN wildcarding
    - QED. Browsers and other systems which act on errors need to probe the resolver
  - 42% of such wildcarding wildcards **everything**, not just `www.*`
  - 6% of sessions (1% of non-OpenDNS sessions) wildcard SERVFAIL
    - QED. Many of those implementing wildcarding aren’t caring to limit collateral damage
  - 5% of sessions (1% of non-OpenDNS sessions) wildcard A records for 0-answer replies
    - EG, a name that is IPv6-only will have an A record wildcard!
    - Since most hosts will do parallel A and AAAA lookups
    - Big problem for IPv6 deployment tests
ISP Manipulation of DNS Results

“Thanks a lot, I will send my ISP to hell now.” -User Feedback

  - Yahoo and Bing lookups return the IP address of a third party controlled proxy
    - All ISPs redirect to an ISP-specific IP within one of two /24 subnets: 8.15.228/24 and 69.25.212/24
  - Google lookups are sometimes faithful, sometimes an ISP-controlled proxy, sometimes a third party proxy

- Systems are clearly a proxy:
  - Proxied traffic appears to be unmodified
  - Not proxied hosts return a redirect to 255.255.255.255
    - Banner reveals the squid version used
  - Invalid HTTP requests refer to “phishing-warning-site.com”
    - A go-daddy parked domain with anonymous registration
    - Only references found with a Google search are complaints about search sites or google apps not working properly
The ISPs and the Mystery:

- Multiple ISPs have this behavior:
  - Cavalier, Cincinnati Bell, DirecPC, Frontier, Insight Broadband, Iowa Telecom, RCN, and Wide Open West
  - Charter: began a partial deployment starting in August 2010
  - Cogent Communications: Appears to be just their small building-scale residential ISP
    - Based on the names of the offending DNS resolvers

- The mystery: *Why* is this happening?
  - What company is selling this service?
  - What do the proxies accomplish?
  - We have attempted to contact representatives of all the noted ISPs: none have responded with answers
Client Transport Issues

- Clients do **not** have unfettered access to DNS
  - 99% can access our server over UDP port 53 but...
  - 10% of clients can’t perform non-DNS: suggesting a protocol-enforcing proxy or firewall
    - Better hope it is correct
  - 1.4% have their request proxied through another server

- Not all DNS types are equal:
  - 98.5% succeed with EDNS0
  - 98.3% for AAAA and TXT
  - 97.2% for unknown RTYPEs (RTYPE=169)
    - 1.2% TXT xor Unknown:
      - If TXT works, unknown will probably work
        - So don’t just shove data in TXT records, it’s OK to do new RTYPES
  - 95.8% for 1400B responses with EDNS0
  - 9% of hosts can’t receive fragments
The NAT’s DNS proxy

- Most NATs have a DNS proxy
  - 73% of the sessions behind a NAT have such a proxy
    - This represents an undercount as initially we only probed for $x.y.z.1$, while 2wire (and others?) use $x.y.z.254$
  - 5% respond to external requests

- These proxies have problems with different types:
  - 96% can do AAAA lookups
  - 92% can do TXT records
  - 91% can do unknown records (RRTYPE 1169)
    - Strongly correlated: only ~1% can do TXT but not an unknown record
  - 91% can process a request which uses EDNS0

- We didn’t test whether the host was configured to use the NAT’s proxy
  - Now we do: the latest version probes the configured resolvers as well as the NAT
  - The latest version also probes for {version, copyright, authors, hostname}.bind, medium and large TXT records, and DNSSEC validation ([www.dnssec-failed.org](http://www.dnssec-failed.org))
Conclusions
“Good luck & good night.” - User Feedback

- Client software needs to work around DNS issues
  - Errors (NXDOMAIN, SERVFAIL) may be changed by the configured recursive resolver
  - The configured resolver may not support all types of requests

- Stub resolvers shouldn’t trust the recursive resolver
  - Perform DNSSEC validation locally
  - Be willing to bypass the recursive resolver entirely
    - The recursive resolver may be slower than direct fetches
      - 19% requires >100ms for a cached lookup

- NAT vendors need to test their software

- Recursive resolvers:
  - Detect and notify/bypass fragmentation issues and TCP failures

- What tests should we add?