Botnet Population and Intelligence Gathering Techniques

David Dagon[1] & Chris Davis[2]

dagon@cc.gatech.edu
Georgia Institute of Technology
College of Computing

cdavis@damballa.com
Damballa, Inc.

BlackHat DC Meeting
2008

Introductions

The Spacious Georgia Tech Campus

based on joint work with:

- *UCF CS:* Cliff Zou
- *GaTech CS:* Jason Trost, Wenke Lee
- *ISC:* Paul Vixie
- *IOActive:* Dan Kaminski
- *Thanks:* Nicholas Bourbaki

DAMBALLA

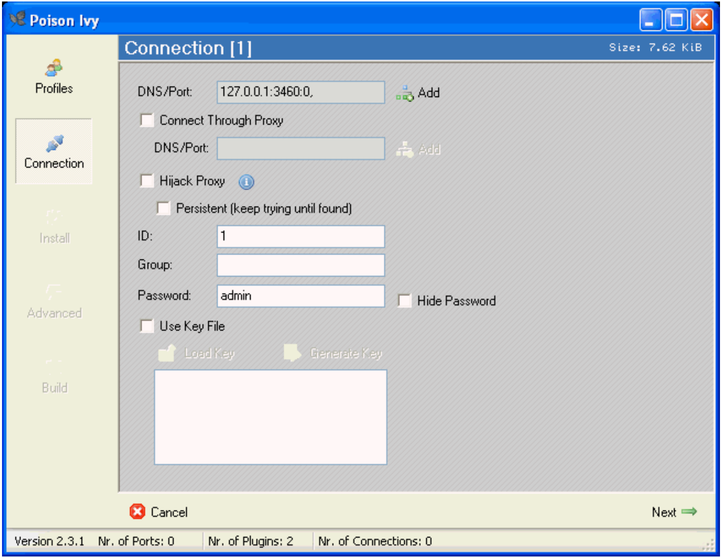
Outline

- Motivation: Infer victim populations with limited probes
- IPID overview
- `BIND` Cache Overview
- Challenges in Modeling
- Solutions
- Further challenges
- Data needs: finding honest open recursives
- Cautions and conclusions

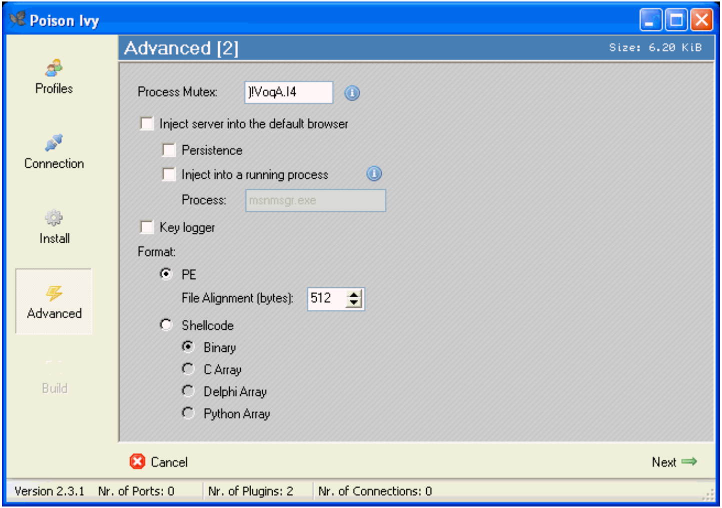
Basic Botnet Facts

1. Most bot malware will utilize domain names so the bot master can move around and the bots can still find him.
2. Many types of bot malware use multiple staged downloads.
3. Many bot masters are just starting to understand how to get their bots to egress from corporate networks.
4. Alot of bot malware is shockingly easy to use

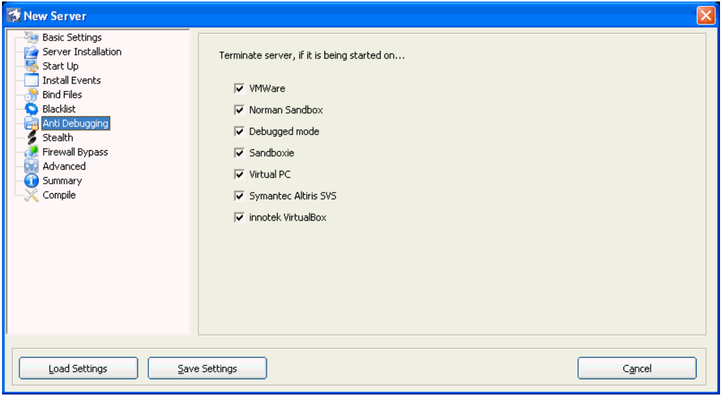
Botnet Basics: Rats



Botnet Basics: Rats



Botnet Basics: Rats



Basic Botnet Facts

1. Not Your Mom's IRC Botnet anymore
2. IRC Botnets are on the decline. Remote Victim Enumeration is becoming harder
3. How do we understand the size and scope of a botnet when we have a limited view?

Understanding IPID

1. Each IP datagram header has an ID field, which is used when reassembling fragmented datagrams.
2. If no fragmentation takes place, the ID field is basically unused, but operating systems still have to calculate its value for each packet.
3. Some operating systems increment the value by a constant for each datagram.
4. Operating systems that increment by one:
    - Windows (All Versions)
    - FreeBSD
    - Some Linux Variants (2.2 and Earlier)
    - Many other devices like print servers, webcams, etc...

Understanding IPID

1. An example of a quiet server:

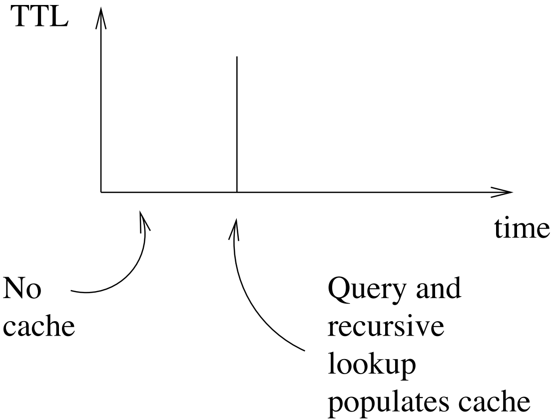
```
cdavis$ hping2 -i 1 -c 5 -S -p 80 XX.YY.ZZ.86
len=46 ip=XX.YY.ZZ.86 ttl=52 id=25542
   sport=80 flags=SA seq=0 win=8192 rtt=42.2 ms

len=46 ip=XX.YY.ZZ.86 ttl=52 id=25543
   sport=80 flags=SA seq=1 win=8192 rtt=48.6 ms

len=46 ip=XX.YY.ZZ.86 ttl=52 id=25544
   sport=80 flags=SA seq=2 win=8192 rtt=48.1 ms

len=46 ip=XX.YY.ZZ.86 ttl=52 id=25545
   sport=80 flags=SA seq=3 win=8192 rtt=43.9 ms

len=46 ip=XX.YY.ZZ.86 ttl=52 id=25546
   sport=80 flags=SA seq=4 win=8192 rtt=42.1 ms
```

Motivation

1. 80% of spam sent via zombies [St.Sauver 2005]; now 90+% [St.Sauver 2007]
2. Volume of phish/malware complaints to ISPs is staggering
   1. Need to prioritize
3. So-called IP-reputation is often merely CIDR-Reputation
   1. DHCP auto-incrementing spam bots, and general lease churn mitigates towards classful scoring, or based on `whois` OrgName or ASN, etc.
   2. Need to remotely assess risk of networks roughly (CIDR) *without relying on remote sensors*.
4. Motivating question: Can we estimate victim populations using simple DNS metrics?

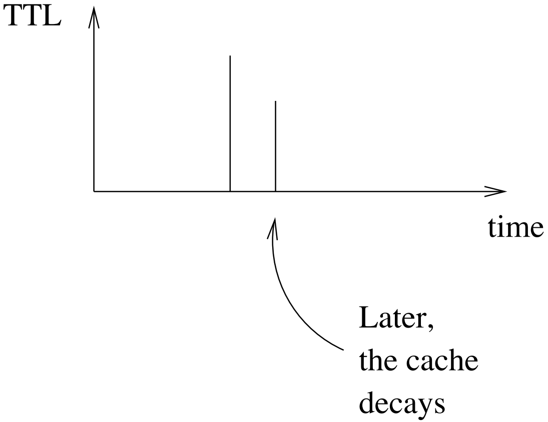
Cache Basics: I

- ***Epidemiological Studies via DNS Cache:***



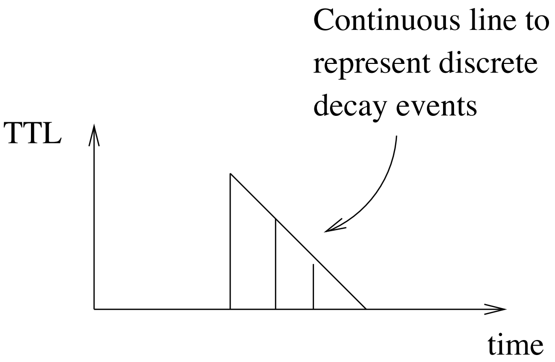
Cache Basics: II

- ***Epidemiological Studies via DNS Cache:***



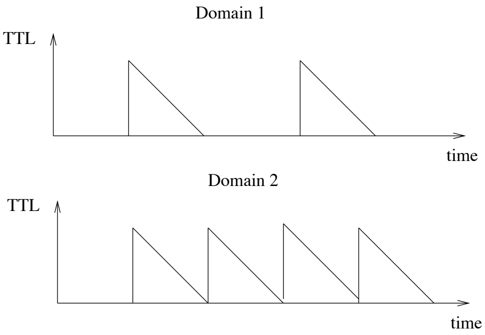
Cache Basics: III

- ***Epidemiological Studies via DNS Cache:***



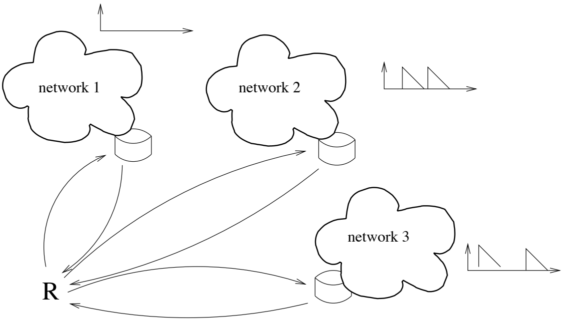
Intuitive Use

- ***Intuitive Difference in Relative Cache Rates***



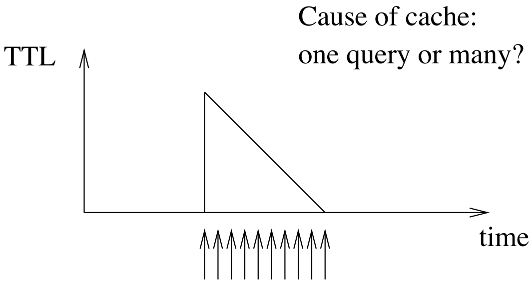
Conception Application of DNS Cache Snooping

- ***Probing Caching Servers for Same Domain***



Problems in Methodology

- ***Caching Inherently Hides Lookups***



Solution: Boundary Estimates

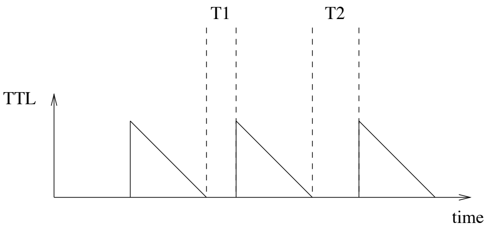
- Assumptions
  - **Property 1**: Bot queries are independent
  - **Property 2**: DNS Cache queues follow a Poisson distribution with the arrival *of uncached phases* at rate λ
    - Note: λ is the "birth process", or arrival rate–the number of events/arrivals per time epoch.
- Are these properties correct?

Independence of Bot Queries

- Two events X_i and X_j, are independent if
  - $P(X_i\ X_j) = P(X_i)P(X_j)$
  - Given the property that $P(B|A) = P(BA)/P(A)$, then to show X_i and X_j are independent, we need to show $P(X_i|X_j) = P(X_i)$
- In the general case, bot victims are randomly selected from potential victims.
- Absent synchronized behavior, one victim's *infection-phase* DNS resolution is independent of any others.
- Example: two victims must visit a webpage to become infected; on a domain TTL-scale, this browsing is independent
- Thus, proptery 1 holds in the general case

Bot DNS Resolution Follows Poisson Distribution

- Does Property 2 hold? Consider:
- ***Intuitive View of DNS Cache Time-outs***



Bot DNS Resolution Follows Poisson Distribution

- The arrival of victims in a queue is trivially modeled as a poisson process
  - This is true of telephony networks, packet networks
  - ...and its generally true of origination from *large* populations of independent actors
- (For some values of large) botnets are large population systems.
- OK, so keep in mind: botnet recruitment that triggers a DNS lookup is a poisson process. We use this point shortly...
- Our current problem: We can only measure cache idle periods however. Are these poisson processes?

Poisson Processes Definitions

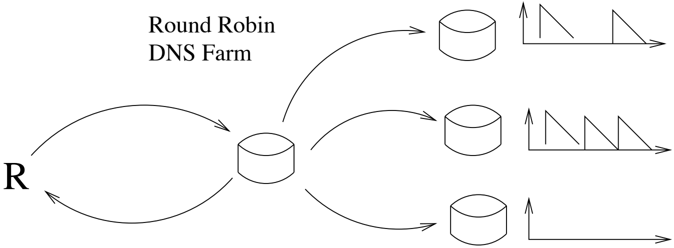
- What's a Poisson process? There are three definitions:
  1. One arrival occurs in the infinitesimal time *dt*
  2. An interval t has a distribution of arrivals following $P(\lambda t)$
  3. The interarrival times are independent with exponential distribution. $P\{interarrival > t\} = e^{-\lambda t}$
- Say, that third definition sure looks like a DNS cache line's idle periods!
- Textbooks then tell used: $\hat{N_{u,l}} = \hat{\lambda_{u,l}}/\lambda$. (There are simple models for deriving populations from arrival rates.)
- Bad joke opportunity: DNS *poisoning* also relies on *poisson* processes

More Problems

- There are hazards in sampling
  - Hidden masters
  - Load balancers using independent caches
  - Policy barriers
- Mandatory
  - Obtain permission and follow RFC 1262 (DNS probes are the spam)
  - Throttle request rates to respect server load balancing (or corrupt data); e.g., 4.2.2.2 throttles non-customers
  - Select small set of suspect domains
- All of these corrupt data collection.
  - (Solutions omitted for space)

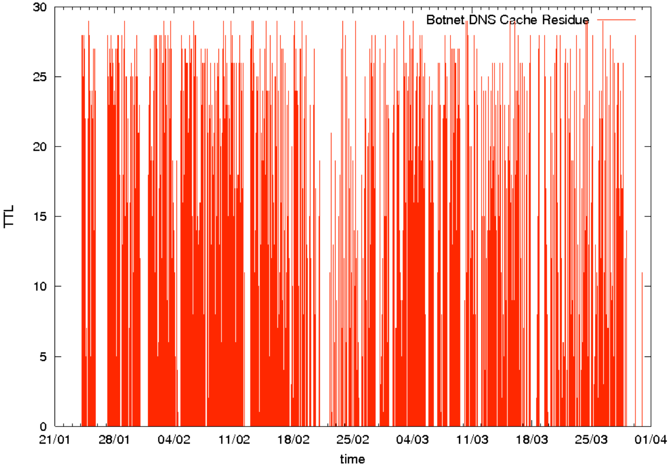
Data Collection Problems

- Sampling is Blind to DNS Architecture



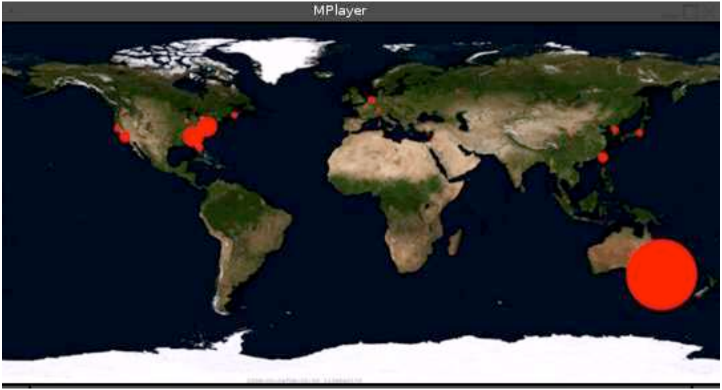
Sample Application

- Study of botnet in Single ISP DNS Cache



Demonstration

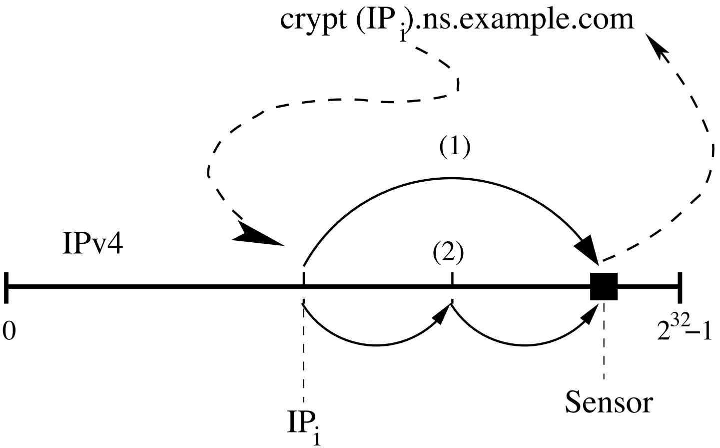
- Plot of output for tracking one botnet (animation may follow)



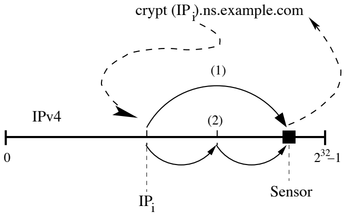
Issue: How to Locate Open Recursives?

- Probing open recursives for domain cache times requires a list of open resolvers.
  - We could just ... scan IPv4 for such hosts
- However, simple queries don't tell us the whole story of the open recursives needed for this task
- We must separate those that are open recursive from those that are open forwarding
- Further, some open resolvers (both full and forwarding) are DNS monetization engines, and don't answer iterative queries truthfully
  - DNS monetization resolvers may not uses caches
  - We wish to identify them, so we can exclude them

One Approach to Recursive/Forwarding Enumeration



Study Methodology

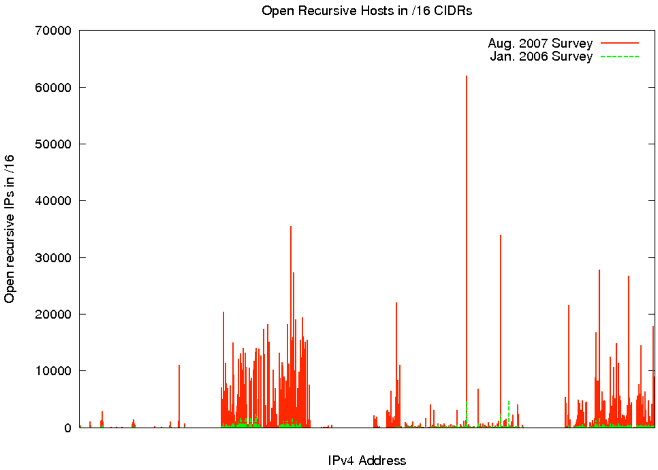


- Unique label queried to all IPv4
- SOA wildcard for parent zone
- Script used to return srcIP of requester
- Logging at NS yields open recursive and recursive forwarding hosts
- Further analysis enumerates “interesting” resolvers

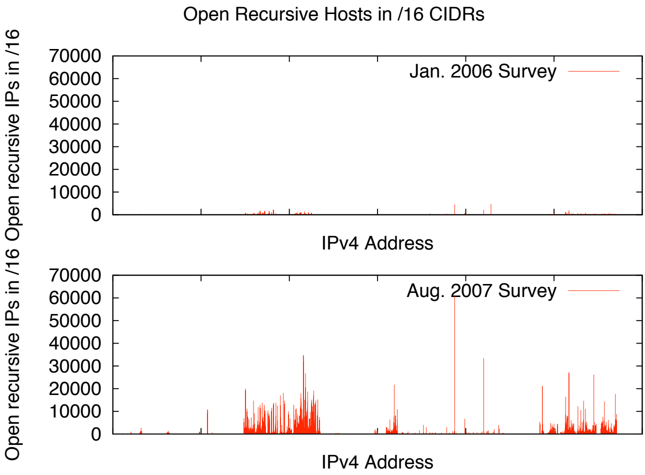
Methodology (cont'd)

- Phase1
  - If response given...
  - Exclude authority open resolvers
  - `fpdns` taken of answering host
  - Perform http request of host
- Phase2
  - Pick 600K open resolvers
  - Ask them repeatedly to resolve phishable domains
  - Note which ones gave incorrect answers
  - If "incorrect", http request to the answered IP

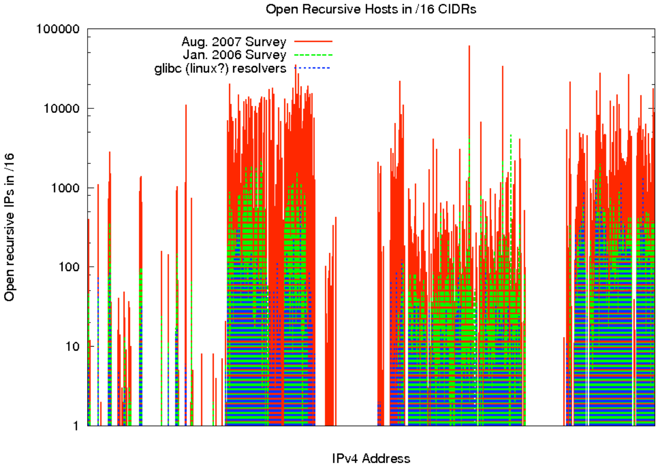
Open Recursion: Comparison of /16s, in IPv4



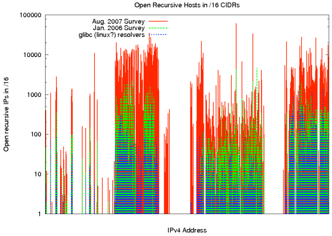
Open Recursion: Comparison of /16s, in IPv4



Open Recursion: Putative GNU libc /16s

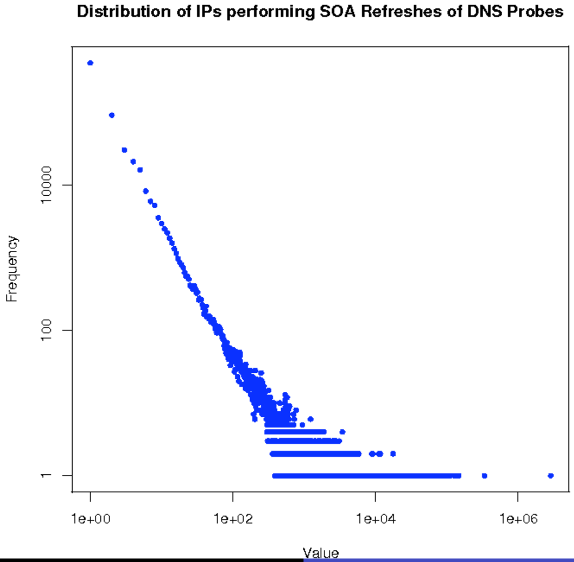


Open Recursion: Putative GNU libc /16s



- gnu libc logic of `AAAA?` $\longrightarrow$ `A?` queries.
- Other heuristics: Windows DNS servers answered authoritatively for queries for `1.in-addr.arpa`,
- Someone needs to update `fpdns` (2005)
- Other "harmless" explanations for open recursion can be considered, and accepted or discarded

Open Recursion: Histogram of Queries to NS



Analysis: What DNS Server is Running?

- HTTP server string fetched from open recursive hosts
  - $\sim$ 20% RomPager, Nucleus, misc. known devices
  - $\sim$ 80% No answer
- Thus, designed study groups:
  - Randomly selected open recursive resolvers
  - Intersection of open recursives and visitors to Google's authority server
  - Intersection of open recursives and Storm victims

Filtering Out "Non-Spec" DNS Servers

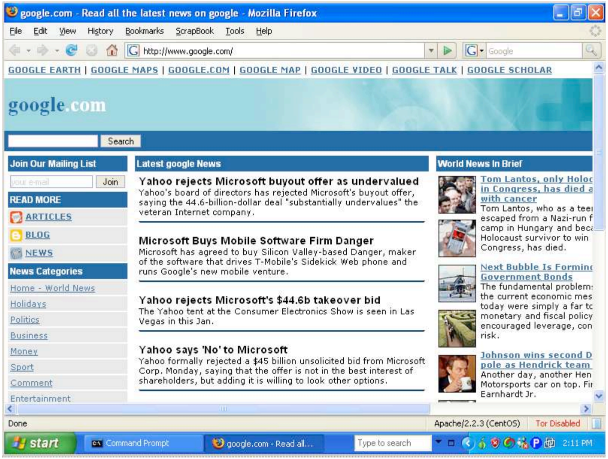
- Methodology:
  - Selected 200K random open recs, 200K open recs contacting Google authority servers, 200K overlap storm
  - Repeatedly queried for "phishable"; 15 min window; 220M probes total over 4 days
  - Diurnal pattern noted (unusual for DNS servers)
  - Approx. 310K-330K resolvers answer; 460K out of 600K total answered
- 2.4% were technically "incorrect" (extrapolates to 291,500K hosts)
- 0.4% were malicious (extrapolates to 68K hosts; 36K measured so far in subsequent full IPv4 sweeps)

Filtering Out “Non-Spec” DNS Servers

- Created database of “proxied” webpages
  - Porn, advertising, and proxied pages(!)
  - $\sim$ 20% proxied/rewrote google.com (demo)
  - $\sim$ 11% proxied a chinese search page
  - $\sim$ 26% proxied a comcast user login
- Methodology reported in `www.isoc.org/isoc/conferences/ndss/08`
- In short, we need to remove these hosts from our open recursive pool

Filtering out “Non-Spec” DNS: Why?

Baaaad DNS (and therefore bad cache timing data):



Conclusions

- DNS cache inspection requires careful analysis
- Merely probing DNS caches alone *does not* reveal victim information
- A model (with safe assumptions) is needed to overcome noise created by variable DNS architecture, events, etc.

Notify, Ask and Coordinate

- Uncoordinated DNS probes pollute IDS logs, generate e-mail complaints
- Use RFC 1262, and common courtesy
- Don't bother checking `mil` or `gov` prefixes

Conclusions

- DNS cache inspection requires careful analysis
- Merely probing DNS caches alone *does not* reveal victim information
- A model (with safe assumptions) is needed to overcome noise created by variable DNS architecture, events, etc.

Notify, Ask and Coordinate

- Uncoordinated DNS probes pollute IDS logs, generate e-mail complaints
- Use RFC 1262, and common courtesy
- Don't bother checking `mil` or `gov` prefixes