Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority

David Dagon[1] Niels Provos[2] Christopher P. Lee[3] Wenke Lee[1]
[1]College of Computing, Georgia Institute of Technology,
{dagon,wenke}@cc.gatech.edu
[2]Google Inc.
niels@google.com
[3]College of Engineering, Georgia Institute of Technology,
chrislee@gatech.edu

Abstract

We study and document an important development in how attackers are using Internet resources: the creation of malicious DNS resolution paths. In this growing form of attack, victims are forced to use rogue DNS servers for all resolution. To document the rise of this "second secret authority" on the Internet, we studied instances of aberrant DNS resolution on a university campus. We found dozens of viruses that corrupt resolution paths, and noted that hundreds *of URLs discovered per week performed drive-by alterations of host DNS settings. We used the rogue servers discovered in this analysis to document numerous live incidents on the university network. To measure this problem on the larger Internet, we generated DNS requests to most of IPv4, using a unique label query for each request. We found 17 million hosts responding, and further tracked the resolution path they used to reach our NS. Unable to find plausible harmless explanations for such a large number of open recursive hosts, we queried 600,000 of these open resolvers for "phishable" domains, such as banks and anti-virus companies. We found that 2.4% of this subsample would reply with incorrect answers, which extrapolates to 291,528 hosts on the Internet performing either incorrect or malicious DNS service. With DNS resolution behavior so trivially changed, numerous malware instances in the wild, and so many other hosts providing incorrect and misleading answers, we urge the security community to consider the* corruption of the resolution path *as an important problem.*

1 Introduction

The Domain Name System, or DNS, plays an essential and often unquestioned role in the operation of networks. We study and measure a growing threat against this service whereby individual infected computers are directed to use "rogue" DNS services instead of those provided by their network. This trend differs from traditional DNS attacks, such as poisoning, since it targets individual users instead of servers. Further, since it involves only the victim and a complicit remote server, the attack is difficult to witness outside of the local network.

We document what appears to be the start of a growing form of attack: the subversion of a host's correct *resolution path*. In this attack, the client is directed to use a rogue DNS server, which provides incorrect answers to queries or selective manipulation of answers for the purposes of commercial gain, phishing or other abuse. In most cases, the users have no indication that the DNS answers are not what the correct authoritative DNS servers would provide.

The attack is by design difficult to detect outside of the local network. Unlike a phishing site, which security researchers can crawl to find, or a botnet, which researchers can measure through flow logs and honeypots, corrupted path resolution leaves little external evidence. In many networks, users are free to select a DNS recursive server of their choice, and network administrators usually lack means to monitor or validate the answers. This openness, and general difficultly in monitoring stub resolver behavior, make resolution path corruption difficult to detect.

Our study of local network traffic confirms the presence of a class of infections that force victims to use remote, rogue DNS services. Our study of IPv4 first revealed approximately 17 million "open recursive" DNS servers. Given their demonstrated use in DNS amplification attacks, this alone has remarkable security implications.

In order to identify hosts actively involved in malicious DNS resolutions, as opposed to those merely misconfigured, we sent repeated queries to 600,000 selected hosts, asking them to resolve various bank, anti-virus, search engine, and other "phishable" domains. Overall we found 2.0% provided incorrect answers [1] (e.g., `NXDOMAIN` answers where `SRV-`

[1]We use the term "incorrect" as a descriptive term, and not to describe

FAIL was the only viable answer, or answers to domains that did not exist). This group was dominated by those commercially altering of DNS traffic, e.g., to display ads or correct typing errors. Further, we found 0.4% provided "misleading" answers, and pointed to proxies. This extrapolates to a population of approximately 291,528 hosts on the Internet that potentially provide either incorrect or malicious answers. While the exact extent is not known, the order of magnitude of such a population supports our central finding: that both commercial and malicious alteration of DNS answers warrants closer study.

These two observations must be read together in context: there are numerous examples of host resolution paths being subverted by malware, and evidence of hundreds of thousands of DNS servers that routinely provide incorrect answers. This pattern, if left unchecked, will lead to the rise of a second, malicious secret resolution authority within the DNS hierarchy. Malware can then trivially change how users experience the Internet: via legitimate recursive servers that correctly surf the DNS zone hierarchy, or via the rogue DNS servers that selectively provide right and wrong answers.

Numerous commercial ventures, mostly focused on correcting user input errors, have conclusively shown the financial value of altering DNS traffic. Our study suggests that malware authors are also aware of the financial value realized in altering normal DNS resolution. There are therefore three classes of DNS answer rewritings: an *opt-in* model, where users are informed of the benefits of DNS answer rewriting, and self-select for the service; an *opt-out* model, where ISPs adopt such technology usually without notifying customers, and let those adversely affected select alternative DNS services; and a *no option* approach, where user settings are forcibly changed, without notice, for malicious purposes.

Specifically, our paper provides the following contributions:

- First, we define the problem of *DNS resolution path corruption*, and rogue DNS service, as distinct from existing DNS abuses (e.g., cache poisoning, fast flux, etc.). We also observe the rise of commercial interests in providing incorrect DNS answers, for the purpose of advertising to customers.

- Second, we describe and demonstrate a technique for measuring the extent to which DNS servers provide either malicious or incorrect answers on the wider Internet.

- Third, we note the implications, for users and networks, of having the current DNS infrastructure supplanted by a malicious authority service, and suggest measures to defend the resolution path taken by users. Since we urge further study of this problem, data from our ongoing survey will be available to the DNS and academic communities, see [1].

the motives behind the incorrect answer. As discussed below, we used other, non-DNS based measurements to further characterize "misleading" answers.

2 Background

We give a brief overview of the Domain Name System (DNS) [18, 19], a critically-important component of the Internet infrastructure, responsible for mapping names to IP addresses. We focus on those aspects of DNS used for the abuse we study. For a general and readable overview of DNS, see [32].

2.1 DNS Operation

DNS is a distributed database that uses a tree structure to organize a namespace, composed of labeled nodes (the root is an empty label). A domain is a node, and fully qualified domains are the bottom-up concatenation of nodes, with each label separated by a period.

A *zone* is a clique of nodes, which form a contiguous tree structure, the top of which is called the *start of authority*, or SOA. The SOA delegates naming authority downward, to *delegation points*, or else terminates with *leaf nodes*. The contents of the SOA are available from DNS authority servers, which typically transmit data to recursive DNS servers, which in turn provide general resolution services for local users. Recursive servers that allow anyone on the Internet to use them are informally called *open recursive*. Such servers are actually one of a set of open resolvers:

1. *Open Recursive* resolvers provide open access to a *full resolver*.

2. *Open Recursive/Forwarding* or *Open Forwarding* resolvers proxy access to a full resolver.

3. *Open Caching servers* have recursion disabled, but still provide access to cached entries.

4. *Restricted Resolvers* provide access to authoritative data.

Each node in a zone contains *resource records* (or RR sets) that contain, typically, mappings between host names and IP addresses. Each RR has a shelf life, or TTL (time to live), measured in seconds, which begins to decay when it is sent from the authority server to caching servers.

DNS clients are varied and many. Typically, operating systems and applications provide limited resolver libraries. Their lack of caches, and typical inability to climb the zone hierarchy to locate records, earn these libraries the name "stub resolvers". Stub code uses the recursive services specified by the host operating system, which is typically set by an administrator, or acquired during a dynamic address allocation session. The recursive resolver used by the stub libraries, together with

the upstream zone access behavior of the full resolver, constitutes a *resolution path*.

Significantly, most operating systems permit sufficiently privileged users to change the default resolution behavior of stub libraries. And some applications, such as web browsers, have their own stub resolvers, which may allow each user to change or proxy resolution behavior.

2.2 DNS Poisoning and Resolution Path Corruption

Since DNS (as opposed to DNSSEC) lacks robust authentication mechanisms, the resolution path taken by a computer is critical to how it experiences the Internet. Typically, whatever answer is provided by a recursive server is implicitly taken as correct by the stub resolver. The lack of authentication, and ability to spoof UDP-based DNS, gave rise to a series of DNS poisoning exploits, e.g., [9, 10, 27], wherein malicious answers were forged and relayed to recursive servers.

Attacks against stub resolvers are typically not called DNS poisoning, since stubs usually lack caching. Usually attacks on stub resolvers involve altering the recursive path used by the host to process all resolution requests. That is, instead of poisoning a cache line in a DNS server, an attack on a stub resolver points the host to a malicious, rogue DNS server.

Figure 1(a) shows a conceptual view of how resolution paths can be subverted for malicious purposes. Users normally request the address of a desired host by consulting a full resolver, such as their network-provided recursive server. The recursive server surfs the zone hierarchy to locate the desired resources, here simplified in Figure 1(a) as a connection to an authority server. A subverted resolution path will cause the infected host to instead use a different resolver, which returns incorrect answers (indicated as `IN A'` in Figure 1(a)). The user is then sent to a rogue site, which may optionally proxy the connection to the original, legitimate site, or perform other deceptions (e.g., phishing).

A few malware samples have altered the DNS resolution path. The `qhost` trojan, for example, altered host DNS settings and browser proxy settings in 2003 [25]. A more recent family of trojans called the `DNSChanger`, did this in only a few lines of code. The recent and wide-spread Zlob [3] attack performed similar DNS alterations.

Perhaps the most famous example of malware changing host resolution behavior was the `zcodec` trojan of 2006. `ZCodec` enticed users to install a "free video player" or codec. In reality, the trojan altered the host "NameServer" registry key, which takes precedence over all DHCP-assigned DNS resolution paths, and directed users to rogue DNS servers. [13]. According to web popularity ranking services, the site used to distribute the `zcodec` trojan briefly rose to within the top 15,000 pages on the entire Internet, over a 3-year average of rankings [4]. The site is still active today.

In this paper, we investigate the degree to which the subversion of resolvers has occurred. Changing a user's resolution path by redirecting her to a malicious DNS server requires the malicious DNS server to be *open recursive*. As such, we are also presenting data from an Internet-wide survey on the prevalence of open recursive servers.

3 Study Methodology

As noted in Section 2, abuses in DNS path resolution are difficult to measure outside of the local network. Since victims contact remote servers directly, the only opportunities to observe the attack are (1) on the compromised host, (2) at the complicit DNS server, or (3) at the local network. We view the first two as difficult and unlikely sampling points, respectively. Observing malicious DNS behavior at the local network level has difficulties as well. Local operators may observe DNS queries leaving their network, destined for remote machines, but absent a policy restricting the use of DNS, they lack direct means of determining whether the traffic they observe is correct.

We therefore measured this problem from both ends: by studying resolution patterns in a local network, and by attempting to find suspicious resolution paths on the Internet at large. For the former, we tapped DNS traffic at a campus border. For the latter, we performed a comprehensive survey of all of IPv4, using a technique that forced open resolvers to contact our nameservers.

By working in both directions, we found evidence of malicious DNS usage on the local network, and found convincing but indirect evidence of this phenomena on the larger Internet.

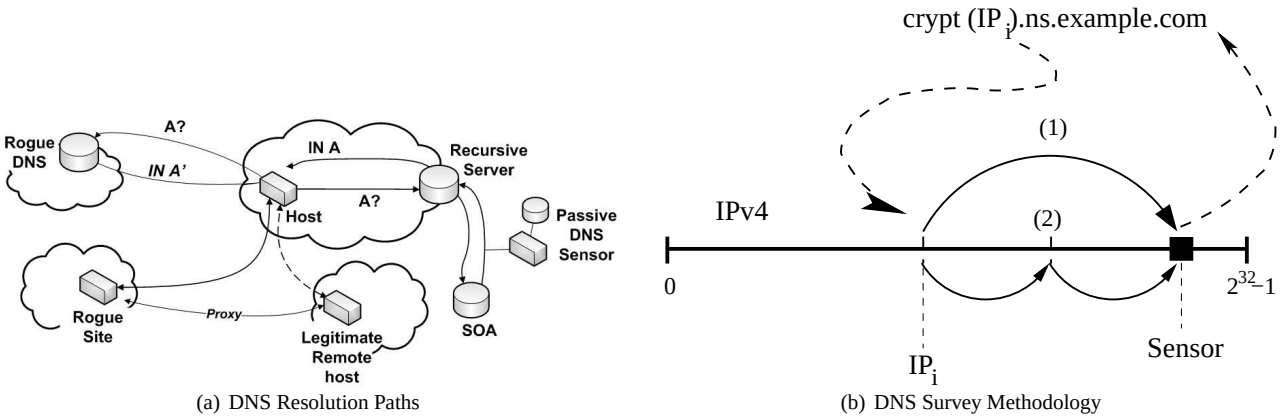
The following sections describe these two approaches in details.

3.1 Local Network Observation

We captured two months of DNS traffic at a busy campus gateway and parsed the packets for the query and resource records. The traffic tap reflected the resolution behavior of approximately 18,000 users. We used a setup similar to Passive DNS Replication [33]. Passive DNS focuses on answer streams directed at resolvers, as shown in Figure 1(a), However, corrupted DNS paths do not transit through local resolvers, and instead make direct connections to rogue DNS servers. We therefore tapped all egress DNS traffic, both UDP and TCP using simple port filter.[2]

During the two month capture period, we gathered 390 GB of network trace files of DNS-only traffic with a total of one billion resource records from 4.5 billion packets. (Since our sample period also overlapped with our DNS probe study, discussed below in Section 3.2, many queries went to non-open resolvers, and were never answered.)

[2]Special care was made to protect the privacy of the students and anyone wishing to reproduce this study would be well-advised to consult their network operators for guidelines



(a) DNS Resolution Paths

(b) DNS Survey Methodology

Figure 1. a) Conceptual view of normal and corrupted DNS resolution paths, indicated as `IN A` and `IN A'` answers respectively. Passive DNS monitoring is also indicated. (b) DNS Survey methodology, revealing host resolution behavior as either (1) open recursive or (2) recursive forwarding.

3.2 Internet-Wide Resolution Survey

We also wanted to measure what we observed in our local network on the Internet at large. This is a difficult problem, since the remote, complicit DNS server potentially controls the entire resolution path. That is, we would not have an opportunity to witness some malicious resolutions at the root, TLD or authority levels, since the rogue DNS server may not even consult the zone hierarchy to find the correct answer. Indeed, in many cases the rogue servers synthesize incorrect answers.

Without the cooperation of numerous network operators, it would be impossible to *directly* measure the extent to which malicious DNS traffic transits on the Internet. We reasoned, however, that since an attacker might not know the IP address of his victim beforehand, the malicious DNS resolver is therefore very likely to act as an open recursive server. That is, they must accept all queries from potential victims, since there is no straightforward way to determine who will ultimately fall prey to a virus.

This reasoning leads us to search for open resolvers, and then identify those that are likely providing malicious DNS services. Our general approach was to pose recursive DNS queries to all of IPv4, for labels that we uniquely controlled.

We started by organizing IPv4 into a series of classful addresses, and excluded non-routable addresses, e.g., [16], using the bogons list published by Team Cymru [28]. We further excluded CIDRs allocated to the U.S. Military and U.S. government. We obtained these addresses by consulting routing prefixes announced by US-government owned ASNs.[3] The remaining addresses were randomly shuffled, and with the ordering of the addresses preserved. This allowed the survey to be stopped and restarted as needed.

For each address, we calculated a label using a hexadecimal representation of a blowfish encryption of each IP. Thus, to a given IP address, IP_i, we asked for an `A`-record for:

$$blowfish(IP_i).\text{parentzone.example.com}$$

For the parent zone of the hashed label (called $parentzone$ in the example above), we used a zone that we controlled, and had delegated `NS` authority. In other words, we asked a unique query to nearly every IPv4 host, regarding a label in our delegated zone. This ensured two properties were maintained in each query. First, each query to each IP would be unique, and therefore (a) not cached by any intermediate server, (b) not easily guessable, even though we did not anticipate problems with spoofed answers in this round of study, and (c) trivially reversible given the blowfish key, so that one can efficiently find which of the billions of IP addresses corresponds to a given label. Second, we ensured that recursive resolutions would be sent to our `NS`, allowing us to see what resolution path was taken by the hosts.

Figure 1(b) shows a conceptual view of how we performed the resolution. Our desire was to not merely enumerate which hosts performed open resolutions, but also find *how* each host looked upward in the zone tree. Figure 1(b) shows two conceptual paths for resolution by the resolver: either (1) directly querying our `NS` for the `A`-record, or (2) forwarding the request to another recursive server. We imagined that the first path might correspond to, for example, a misconfigured nameserver that was also open recursive. The second path represents an open forwarder, as discussed in Section 2, such as a home computer that forwarded DNS requests to the ISP's DNS servers.

[3]Based on previous experiences with DNS surveys, we found that sending DNS queries to these networks invariably resulted in questions forwarded to our campus network abuse group. We reasoned that such carefully watched networks were unlikely to have open resolvers.

The other two types of open resolvers, open cache and authoritative, were not relevant, and so we did not design a probe technique to map them specifically. Open cache resolvers, for example, do not provide recursion, and authority servers only return records for their zone. That is, of the four types of open resolvers noted in Section 2, only the first two were deemed to be useful for studying resolution path corruption.

Thus, in our study, the sensor in Figure 1(b) examines the DNS query traffic and for each query compares the look-up string, i.e., $blowfish(IP_i)$ with the IP address, IP_j, that sends the query. This would tell us the DNS resolution behavior of IP_j. Specifically, if the $IP_i = IP_j$, then IP_j is an open recursive server; otherwise, it is recursive forwarding. If IP_j is properly configured (i.e., non-open recursive), it would have dropped/ignored our query and as a result, our sensor will not receive a look-up for $blowfish(IP_i)$.

Another simple technique to separate forward recursive servers from other open recursive servers is to reply with the IP_j as the answer at the sensor. When the answer comes back from IP_i, check if the answer matches IP_i. If it is forwarding, then the answer will not be IP_i, but rather IP_j. This approach requires less processing than encrypting IP addresses, but is more susceptible to spoofing.

We performed two studies of IPv4. In our second scan of IPv4, once we determined that an IP address functioned as an open recursive resolver, we performed additional queries to fingerprint the implementation of the DNS server. The three additional queries we sent were a query for `version.bind` in the `CHAOS` name space, a truncated query where we set the `TC` bit and an `IQUERY` that is a deprecated way of performing a reverse lookup.

In our second pass, to avoid state for randomly ordering the IP addresses we probed, we employed a variable-sized block cipher, as done in [23]. Determining the next IP address simply involves incrementing a 32-bit counter and encrypting it with the block cipher. The block cipher essentially gave us a permutation on the 32-bit IPv4 address space.

We created fingerprints for each resolver from the response packets according to Bernstein's methodology [11], which consists of concatenating the codes described in Appendix A, Table 2 depending on fields and flags in the DNS response.

In addition to the DNS queries, we also performed an `HTTP GET` request against the IP address of the open resolver to determine if it was also running an HTTP server. A successful `GET` request allowed us to analyze the HTTP headers for information such as the server version and the timezone the server ran in. We pretended to be a version of Internet Explorer, to more fully interact with HTTPD-enabled bots that screen requests based on the user agent [14].

4 Analysis

Our study focuses on a class of attack that changes the resolution path of hosts, and directs users to malicious DNS servers. Although some anti-virus vendors report in depth on this problem, (in particular, see the analysis in [12, 15]), we wanted to find local evidence of resolution path corruption. We then applied our DNS scan technique to see if we could find evidence of this abuse in the wider Internet.

4.1 Evidence of Local DNS Abuse

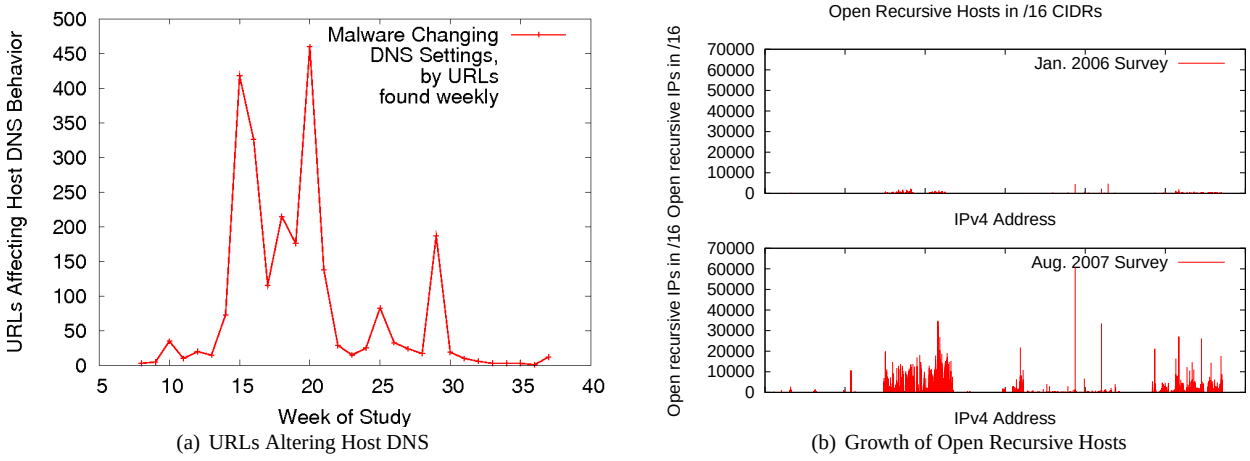
The data collected at the university core router was enormous. In [37], the authors were able to use hand sifting and simple threshold analysis to spot several common trends. For example, they sorted alphabetically the domain query strings, and easily found typo squatting domains. Likewise, they sorted domains by the number of `A`-records to find fast flux [29] behavior. We observed these behaviors as well, but our task required us to find much smaller needles in the haystack.

We used two approaches to locate path corruption examples. First, we matched queries directed to open recursive DNS servers and Storm Worm [2] infected nodes. Thus, when university users queried remote open recursive server or known bots, we could use this as a low pass filter, and further analyze the matching traffic. Second, we also obtained and ran DNS altering malware, to learn the location of rogue DNS servers. We then used the IPs of the rogue DNS servers to bootstrap the search for infected hosts.

To identify malware samples that change DNS settings, we took 194,372 virus execution traces from the Malfease project, `https://malfease.oarci.net`, and examined the underlying execution trace, which includes system calls, to see if the malware altered windows registry keys affecting DNS resolution. We found 6 samples that change the "NameServer" registry key. We further performed a search on Malfease for malware identified by AV tool scans as affecting DNS, e.g., the `Trojan.DNSChanger` virus. While hardly complete, since this depends on the non-standard naming conventions of anti-virus vendors, it helped characterize the samples.

There were too few `PE32` binary samples that changed the DNS path to reveal any trends. So we consulted the malware detection infrastructure used by Google [24], and looked for web pages that, when visited, would cause hosts to change their DNS resolution settings (e.g., by using web exploits to alter the host NameServer registry key). Over the last six months, we found 2,107 such pages distributed over 605 domains. In total, these pages pointed hosts to 75 unique remote DNS servers. The graph in Figure 2(a) shows the number of URLs encountered weekly is quite high. Where there may be other propagation vectors we did not measure in this paper (e.g., spam traps), the web and binary analysis suggests that this form of attack is current and prevasive.

The trend we observe is that, after the initial DNS-altering virus in 2003, numerous virus samples started to appear in 2005 that changed host resolution paths, e.g., `DNSCharger`. These virus-driven changes were supplemented or replaced with web-based exploits that performed the same. This attack



(a) URLs Altering Host DNS

(b) Growth of Open Recursive Hosts

Figure 2. (a) URLs encountered per week that alter host DNS settings. (b) Prevalence of Open Recursion in IPv4 /16s, January 2006 and August 2007 compared.

exposes the victim to identity theft, without the need for elaborate host-based keylogging or rootkits. All of the malicious behavior exists on remote servers, made all the more agile by the use of a rogue DNS server.

To bootstrap our analysis, we executed 8 samples of DNS changing malware in honeypots, and observed the changes made to the honeypot's resolution settings. This yielded a set of 8 different IP addresses pointed to by the viruses. We reasoned that traffic to/from those IPs would be most likely malicious, and used this as a filter for our trace files. We checked our DNS data collection, and found numerous instances of DNS traffic sent to these remote sites from the campus network. In several cases, a remote DNS server in Russia served as the primary recursive DNS server for several compromised machines on the US-based university network. Specifically, there were several instances of campus hosts using known malicious resolvers from the Ukraine region for queries like `time.windows.com`, `www.amazon.com`, `www.facebook.com`, and `sb.google.com`. The queries were blocked from reaching the servers, so we did not learn what answers were provided. Below, we describe how we generated queries to other, non-blocked rogue DNS servers, and documented incorrect answers.

4.2 Understanding the Nature of Open Recursives

Despite the relatively small numbers of malware samples that altered DNS settings, we nonetheless found numerous infected individuals in an average-sized university. This suggests there may be a wider prevalence. While our evidence of local malicious DNS traffic caused by infections was quite clear, finding a similar pattern of abuse on the wider Internet is not as straight forward. As noted in Section 2 it is nearly impossible to directly observe. Our recursive probe technique, however, gave us a starting point for indirect evidence.

Our general analytical approach was to take the set of open recursive servers, and attempt to find which ones were used for malicious DNS services. We applied a series of filters (e.g., removing linux hosts, and embedded DSL devices), in order to better locate malicious resolvers.

Our late August 2007 scan found 10.4 million open resolvers, while our early September 2007 scan found 10.5 million resolving hosts. The union of the two sets yields 17,365,759 open resolvers, since only 3.6 million IPs were in common. The scans were only a few weeks apart, suggesting a mass migration of some 7 million DNS server addresses.

The August and September numbers were also a significant increase over a January 2006 scan that found only 634,941 hosts. To illustrate the IP diversity gained by this increase, a plot of the January and August data appears in Figure 2(b), where the IP address is plotted on the x-axis, and counts of open resolvers by /16s form the y-axis. The graph high points all correspond to ISP allocations.

We note that according to site ranking services, the `zcodec` malware propagation site reached its peak popularity (a top $15,000$ site) in Q3 of 2006–months after our initial survey. By looking at the reports of the gross numbers of open resolvers found by others (e.g., [35], who reported a spike in open recursive hosts in July, 2007), we theorize that just in the last year, there has been a dramatic rise in the number of open resolvers, on the order of several million. This by itself has profound implications for security, given the role these machines could play in denial of service attacks using DNS amplification [31].

Clearly not all open resolvers provide malicious DNS services. With tens of millions of open recursive hosts observed,

we endeavored to find harmless explanations for the open recursive behavior in our data. We first theorized that some of these open recursive hosts could be hobbyist machines or open source DNS servers used by small businesses. Linux machines could of course have their resolution paths changed, but we found it unlikely the `Win PE32`-based binaries corrupting host resolution would act on Unix machines, other than in specialized emulation contexts.

To test this explanation, we analyzed the pattern of resolutions used by hosts. When performing recursive lookups, Linux hosts generate a distinct pattern of `AAAA` queries, followed by an `A` query, because of the forward IPv6 compatibility logic in glibc, `glibc-2.6.1/resolv/gethnamaddr.c`. That is, Linux hosts perform an IPv6 lookup (regardless of whether there's a non-link-local v6 interface), and then an IPv4 lookup when the query fails or the querying host has no 6-stack. As a result, when we observed a host performing the resolution pattern (`AAAA` then `A`) to our `NS`, we deemed it to be a Linux host. This heuristic will have diminished value when Vista's stub resolver is refined.

We found that only $169,407$ of the open recursive hosts used one of $37,429$ unique Linux forwarding resolvers. Filtering our list of open recursives this way did not provide a satisfying explanation for the large number of resolvers. There had to be other factors behind these open recursive hosts.

We next theorized that many of the open recursive hosts were running a DNS server embedded in a home networking appliance, such as a premium DSL router. Since we also sent a web request to open resolvers, we checked for hosts answering with server strings that correspond to embedded devices (e.g., RomPager, Agranat-EmWeb). Here, the theory was that these embedded devices, while perhaps poisonable using traditional DNS cache attacks, were less likely to be used as a resolving authority for malicious purposes. A break down of the server strings appears in Table 7. We found a total of $417,327$ such hosts–again too few to explain the surge in open recursive servers.

We next looked at properties of the recursive servers themselves, and how they resolved our probes. As noted in Section 3, we tracked the IP address of the open recursive resolver that we asked to resolve a query and the IP address that eventually contacted our authoritative name server for that query. We recorded approximately 23 million such pairings. About 96.4% or about 15 million resolvers forwarded their query to another resolver, whereas only about $580,000$ resolvers contacted our name server directly (open recursive) instead of forwarding the request (open forwarding). We analyzed the number of IP addresses and $/24$s behind each forwarding resolver. Approximately 71% of forwarded resolvers have only one open recursive resolver that forwards to them and 87% of all resolvers have only a single $/24$ that forwards to them, and shown in Figure 3.

Some resolvers received recursive forwards from over $20,000$ different $/24$s. The resolvers with a large number of forwards from open recursive servers are mostly located in China, Korea and the US (Table 4b). The network and geographic diversity found in these IPs was curious, since it was not clear what relationship existed between so many open resolvers, and other hosts in remote countries and networks. The actual recursive forwards with the most clients using them are located in Italy, Netherlands, and the United States. The top 10 recursive forwarding servers are listed in Appendix Table 3. We found that a high percentage of hosts using a particular forwarder are in the same country as the forwarder.

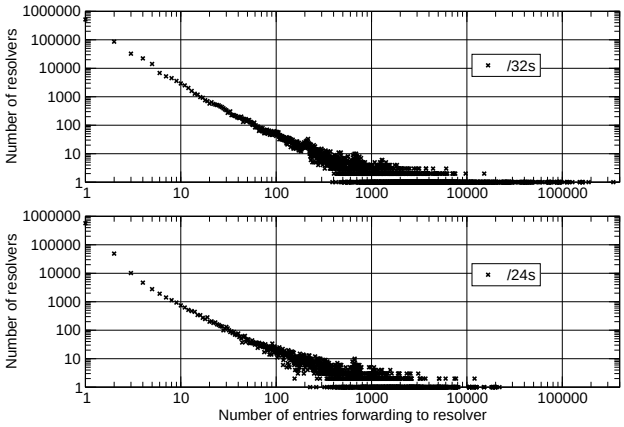
Although, approximately half of the legitimate DNS servers are configured correctly to be not open-recursive, the forwarding tables we built allowed us to query even closed resolvers. Any open-recursive server that forwards its queries to the closed resolver can be used to query the closed DNS server. Although there is a chance that the open-recursive might be lying, we can probe many of them and then do a majority vote on the answers.

We continued to look for explanations about the behavior of these open recursive servers. So we next compared them with two different sets of known recursive resolvers. The first set was a subsample of DNS servers resolving Google domain names, totaling some $900,000$ IPs over a period of three months. Essentially, this was a sampling of IPs that consulted the authority servers for `google.com`–DNS servers refreshing cache entries. The other set consisted of about $80,000$ IP addresses contained in the "glue records" for the `.com` TLD. Essentially, these were the IP addresses of nameservers listed in the `.com` zone.

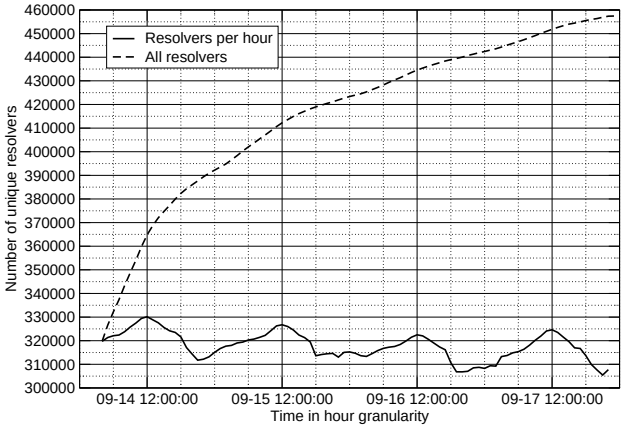
Since these hosts were already known to be DNS servers, we sought to explain what fraction of the 17 million open recursive servers were observed to be DNS servers in other contexts. We found about 50% of the glue IP addresses in our set of open recursives and about 25% of the Google resolvers. The overlapped hosts are likely DNS servers, misconfigured to be open recursive. There remained, however, a very large number of open recursive hosts, numbering in the millions, whose role as a DNS server cannot be explained. The fact that they did *not* consult the authority servers for `google.com` suggests they forward recursive queries to other machines.

Unable to find other plausible harmless explanations, we instead began to look for *negative* explanations of the hosts behavior. We consulted the "BL history" of each IP. We built a database of every black listed host noted by SpamHaus [26], for a period of six months prior to our study. We used SpamHaus's "XBL", or exploits block list as a reference for IP reputation, since the XBL identifies IPs hosting or sending malware (e.g., viral attachments in email). Hosts are delisted automatically after a few weeks, or by request.

Figure 4 shows the distribution of $396,000$ open resolvers in our study that had a negative BL history. Many had short listings–often a single event, while others had persistent, mutli-week listing times. This is fairly typical of dynamic



(a) Forwarding Resolvers and /24s



(b) Diurnal Reachability of Resolvers

Figure 3. (a) the number of resolvers being used to resolve forwarded query and how many open recursive IP addresses or $/24$ networks are forwarding to them. The distribution follows Zipf's law. (b) A diurnal pattern of hosts probed over time, suggesting home users acting as resolvers.

hosts that SpamHaus tracks. We similarly noted the overlap between our open resolvers and the Storm Worm-infected hosts. We obtained a list of Storm bots, using a variety of data sources. [14, 30]. We counted $754,159$ hosts in the Storm botnet that were open recursive.

5 DNS Servers That Lie

Thus far, our analysis generated a set of open resolvers, and noted (a) the remarkable migration of 7 million hosts between scan events, (b) the lack of any satisfying explanation for why these hosts are otherwise open resolvers (e.g., authority servers, embedded devices, Linux machines). To find the hosts in this set that were being used for malicious resolution purposes, we designed a second study.

We selected a set of approximately $600,000$ resolvers that were chosen from three different categories: $200,000$ selected uniformly randomly from all $17,000,000$ resolvers, $200,000$ selected from resolvers that overlap with resolvers contacting Google, and $200,000$ IP addresses selected from known Storm bot infected nodes. Over a period of four days, we asked these IP addresses to resolve 84 different domains. The domains consisted of a subset of banking sites, social networking sites, anti-virus sites and other domains likely to be a subject of Phishing attacks. We sent about 670 probes per seconds so that we would probe each resolver every 15 minutes on the average. Over the course of the four days, we sent approximately 220 million probes.

To determine if a resolver provides incorrect answers, we identified the set of authoritative net blocks that contain valid responses to `IN A` queries for every single domain. We then compared the answers we received from the probed resolvers and checked if the answers fell within the netblocks we identified. For each probe, we recorded the time it was sent, the answers we received or a `timeout` in case we did not receive any answers. Note that we use the term "incorrect" in a precise clinical sense, and do not merely use DNS to describe the result as "false", "misleading", or "malicious". Instead, to characterize the motives behind the incorrect answers, we visited the IPs provided by the DNS server.

Figure 3 shows how many unique resolvers answered our probes per hour and also the cumulative number of resolvers that answered our probes so far. The graphs show that over the course of our study we received answers from approximately $460,000$ of the approximately $600,000$ resolvers we probed. However, at any given hour, we received answers from only about $310,000$ to $330,000$ resolvers. In general, we would expect that DNS servers to be available all the time. However, as seen in Figure 3, 30% of all answering resolvers are unreachable at any given time. Also, the small diurnal trend we observed seems to indicate that the majority of machines we probed might be end-user devices or hosts that are being turned on and off depending on their usage.

Table 1 shows statistics for the resolvers organized by country. The country was derived from geo-location data on the IP address of the resolver. We use four different categories to describe the nature of an open-recursive resolver: `All True` indicates a resolver that answered correctly to all our queries, `All lies` indicates resolvers never correctly answering queries, `NXDomain` indicates resolvers returning addresses to `IN A` queries for non-existent domain names, and `Buggy` for resolvers that return `IN A` records that are off-by-one from the correct answer in one of the octets of an IP address.

A resolver that never correctly answers a query is often in-

Country/Type	Number of resolvers	Answering	All True	All Lies	NXDomain	Buggy
All	593092	457643 (77.2%)	446689 (97.6%)	566 (0.1%)	9955 (2%)	212 (0%)
Storm	211778	166869 (78.8%)	164107 (98.3%)	86 (0.1%)	2560 (2%)	26 (0%)
Google	192629	160385 (83.3%)	158129 (98.6%)	105 (0.1%)	1829 (1%)	142 (0%)
Random	188685	130389 (69.1%)	124453 (95.4%)	375 (0.3%)	5566 (4%)	44 (0%)
Other Fingerprints	319684	227007 (71.0%)	**220143 (97.0%)**	422 (0.2%)	**6244 (3%)**	90 (0%)
Fingerprint: 1q1	141490	123747 (87.5%)	122052 (98.6%)	70 (0.1%)	1482 (1%)	102 (0%)
Fingerprint: ttt	107049	89795 (83.9%)	87484 (97.4%)	72 (0.1%)	2149 (2%)	20 (0%)
Fingerprint: 5q5q	24869	17094 (68.7%)	17010 (99.5%)	2 (0.0%)	80 (0%)	0 (0%)
Unknown OS	553956	423212 (76.4%)	412443 (97.5%)	556 (0.1%)	**9796 (2%)**	207 (0%)
RomPager	31260	26733 (85.5%)	26572 (99.4%)	4 (0.0%)	149 (1%)	3 (0%)
Linux	7876	7698 (97.7%)	7674 (99.7%)	6 (0.1%)	10 (0%)	2 (0%)
USA	127008	101851 (80.2%)	97196 (95.4%)	293 (0.3%)	4327 (4%)	11 (0%)
Turkey	57041	48593 (85.2%)	**48581 (100.0%)**	1 (0.0%)	4 (0%)	7 (0%)
Brazil	30900	24876 (80.5%)	24850 (99.9%)	5 (0.0%)	13 (0%)	2 (0%)
Spain	30458	21867 (71.8%)	20070 (91.8%)	2 (0.0%)	1794 (8%)	0 (0%)
Japan	21370	16758 (78.4%)	16737 (99.9%)	4 (0.0%)	13 (0%)	0 (0%)
India	17611	16094 (91.4%)	16054 (99.8%)	1 (0.0%)	20 (0%)	1 (0%)
Peru	16414	15890 (96.8%)	15878 (99.9%)	2 (0.0%)	9 (0%)	1 (0%)
Thailand	15954	14640 (91.8%)	14513 (99.1%)	28 (0.2%)	68 (0%)	6 (0%)
China	28683	13398 (46.7%)	10920 (81.5%)	38 (0.3%)	**2406 (18%)**	20 (0%)
France	19317	13137 (68.0%)	12893 (98.1%)	9 (0.1%)	236 (2%)	1 (0%)
Italy	16984	12292 (72.4%)	12248 (99.6%)	7 (0.1%)	30 (0%)	0 (0%)
Taiwan	6158	4162 (67.6%)	4004 (96.2%)	12 (0.3%)	15 (0%)	**134 (3%)**

Table 1. The table shows the geographic distribution of probed resolvers and how they answered to probing queries. The table also shows statistics for the operating system or fingerprint class a resolver belongs to.

dicative of captive portals where users need to authenticate before they can use the Internet. Making these resolvers accessible over the Internet is likely due to misconfiguration. It is interesting to note that Turkey has the largest fraction of resolvers that return accurate answers. The country with the largest fraction of resolvers answering for non-existent domains is China.

In addition to looking at the geographic distribution of resolvers, we also analyzed them according to the sample set they belonged to: Random, Storm or Google. There are no significant differences other than the fact that the randomly sampled set has a larger fraction of resolvers that answer for non-existent domains.

We also separated the resolvers into different classes depending on their DNS fingerprint and the operating system implied by the HTTP Server header. We notice that resolvers running Linux web servers have much higher availability and more correct query answers compared to the set of resolvers for which we could not determine an operating system version.

To further characterize the "incorrect" answers, we accepted the answers given by the DNS server, and browsed to the web site they resolved. Thus, we visited sites such as Ebay, Amazon, and Google, using the DNS server's incorrect answer. We captured each of these pages into a database and extracted them later for analysis. By hand analyzing over 250 randomly sampled webpages, we found the common types of misdirection and built heuristics to detect them. A large number of sites were parked domain splash pages (although the real domain does exist) with 221 pages for one domain, 224 pages for another, and 96 for yet another. We also found 48 proxied google pages, 29 Chinese splash sites, and 66 Comcast pages requesting a completion of registration. These ratios of parked, proxied and apparent phishing pages held over the entire database.

All of these pages, of course, could be altered trivially by the proxying host. And all of them let the remote site act as a man-in-the-middle for all transactions (checking mail, logging in, searching etc.) In Table 1, we identify such DNS answers as "lies", since they point to pages that are clearly not the original requested resource (e.g., Ebay), but provide no indication to the user that they are not associated with the site the user looked up.

5.1 Commercial Abuse of DNS

Companies such as Nominum, Paxfire, Barefruit, Simplicita, and OpenDNS derive commercial value from altering some DNS answers. The primary motivation is a practice informally called *error-path correction* in which bad user input errors lead to DNS queries that should normally return NXDOMAIN. Instead of forwarding NXDOMAIN to the end host, the DNS resolver returns IN A records to an IP address that return advertisements and search results relating to the incorrectly entered host name. In some cases, the commercial resolvers also return incorrect IN A records when a DNS query has timed out. In the case of OpenDNS, the user is prevented from resolving known malicious (phishing) domains. This practice is the dual opposite of the involuntary, malicious path corruption attacks noted in Section 4.

Our analysis shows that approximately 2% of all resolvers answer for non-existent domains. Most commercial DNS services behave this way as well. This has unfortunate consequences for non-HTTP protocols such as SMTP where mail is being delivered to the IN A records if no IN MX record can be found. In China, this practice is most prevalent with about 18% of all probed resolvers answering for queries to non-existent domains.

Thus, these DNS services use either an opt-out or opt-in system to affect user DNS settings. This contrasts with the "no option" model used by malware, which of course does not notify users. We see consent and user notification as the key issues in all of the DNS-altering systems (both commercial and malicious). Of all the commercial DNS services, only OpenDNS appears to have a voluntary system, coupled with meaningful user notification and education.

We note that there are no RFCs, policy guidelines or even informal standards to guide DNS rewriting. Given the widespread use of these commercial services (and the parallel rise of malicious DNS answers), we urge further study of this area.

5.2 Implementation Errors

We found a noticeable number of resolvers that returned incorrect answers due to implementation errors. Although we have no insights into the nature of the bug, the behavior was deterministic and happened only for queries that return multiple IN A records. In some cases, resolvers decremented the second-most-significant octet in one of the IN A records, in other cases, we found the least-significant octet decremented. The approximately 200 resolvers we found behaving this way either timed out to our CHAOS queries or answered with $9.3.1$ or $9.4.1$ indicating the version of BIND they claim to run.

6 Related Work

The closest work to ours is [37], which used passive DNS monitoring to observe numerous resolution anomalies such as typo squatting. By sorting epochs of DNS traffic, and noting the IN A's geographic origins, the authors were also able to identify fast flux domains. Their description of fast flux is more narrow than the Honeynet Project paper, [29], which takes a general view of flux, noting that it may involve both DNS indirection via a rotating NS layer, and an HTTP proxy layer. The double-flux described in [29] is a single example of the misbehavior we describe. Our work considers *resolution path corruption* as a general form of attack, which may involve the use of rotating malicious NS servers, as well as malicious trojans to alter a victim's default recursive behavior, and ultimately, the creation of a second malicious resolution authority.

Part of our analysis of course makes use of passive DNS replication, first introduced by Florian Weimer [33]. Technologically, we merely used a datastore technique similar to Weimer's. From a policy point of view, however, logging DNS traffic *not* flowing to or from known DNS servers, as in [33], has tremendous privacy implications. For this reason, we have not proposed a general extension to passive DNS, and leave this for future work.

Scanning large portions of IPv4 for DNS activity was addressed in [20], where the authors considered how malicious reverse DNS probes can reveal darknet space. Their work endeavored to better mask darknet space, while ours endeavored to discover hosts in routed space.

Our work is different from those studying DNS cache poisoning. Our analysis focused on the host-based manipulation of stub resolvers, and the answer stream from selected DNS servers. In contrast, surveys such as those by the MeasurementFactory [34] examined poisonous answers (usually from authority servers) for parent zones (e.g., TLDs and the root zone). Such surveys point to misconfiguration, while our study points to intentionally altered DNS answers, either for commercial or malicious reasons.

Our works fits into the larger set of literature that characterizes DNS behavior. In this vein, open recursion has been studied as a security problem on the Internet. [31]. Our survey also revealed likely misconfiguration of DNS servers. Our concern was on the intentional malicious subversion of DNS; for a treatment of how general configuration errors affect the robustness of DNS, see [22]. For a thoughtful treatment of open recursion, see John Kristoff's assorted talks on the subject [17].

Our survey also noted several DNS deployments that superficially appeared vulnerable. The vulnerabilities of various DNS systems have been observed since [10]. Some portions of our analysis relied on historical information associated with IP addresses, which may have been affected by DHCP churn. In [36], the authors addressed this issue directly.

7 Conclusion and Future Work

We have witnessed an increase in malware that changes host resolution paths. This trend, combined with a large supply of open recursive hosts, threatens to create a new, malicious second authority within the DNS hierarchy. We urge the attention of the community to the following issues.

Measurement. Our short study provides a glimpse into a group of tens and hundreds of thousands of DNS servers that provide incorrect DNS replies. We need to better understand and detect when this is done for commercial gain, with varying levels of transparency and notification, and when this is done for purely malicious purposes.

DNSSEC/DLV. We believe DNSSEC [6–8] provides a solution to malicious path changes (and other issues) if end-to-end validation is permitted on hosts. It remains to be seen what manipulations malware can have on the host's use of the validation process.

As reported in a recent study [21], DNSSEC deployment requires good understanding of managing cryptography, e.g., key management, and coordination across administrative domains, and support of gradual roll-out (e.g., supporting the DNSSEC in isolated "islands"). These are non-trivial issues to overcome and will take time before DNSSEC is fully deployed on the Internet. This suggests that DNSSEC Lookaside Validation (DLV) [5] records may play an important role as well.

We are also concerned that the monetization of DNS answer rewriting may provide a counter-incentive to the adoption of DNSSEC. For example, networks monetizing user typos may have financial reasons to discourge the use of end-to-end DNSSEC, which would reveal the substitution of NXDOMAIN answers.

Blocking. It seems likely some networks will impose simple restrictions on egress DNS traffic (e.g., as many `.aero` zones do already) and require the use of local servers. Some networks may allow the use of remote DNS servers; however, as our study has shown, that trust may be misplaced. If users do not consent to or know about DNS alterations (either from commercial DNS services, or malicious software), then blocking user DNS traffic at the edge may provide an appealing solution to some operators, particularly enterprises.

The security community needs to understand how this might create a brittle DNS infrastructure, and what tradeoffs exist in local networks.

Recovery. When rogue DNS servers are taken down or blocked, the victims are left without DNS, and ISPs may face enormous support costs. It is essential that the security community coordinate with the ISPs and law enforcement.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants CCR-0133629, CNS-0627477, and CNS-0716570, and by the U.S. Army Research Office under Grant W911NF0610042. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation and the U.S. Army Research Office.

The authors gatefully acknowledge David Presotto and Ankur Jain of Google for assistance with the Google DNS data, as well as Panayiotis Mavrommatis and Dean McNamee of Google for help with the malware detection infrastructure. The authors also gratefully acknowledge the draft review and advice provided by Paul Vixie, David Ulevitch, and Cricket Liu. Expert system administration was provided by Robert Edmonds. And endless supply of patience and `abuse@` support was provided by the Georgia Tech Office of Information Technology. Peter Honeyman and M5 Hosting also assisted with DNS scanning.

References

[1] Operations, analysis, and research center. `https://oarc.isc.org/`, 2007.

[2] Storm worm. `http://en.wikipedia.org/wiki/Storm_Worm`, 2007.

[3] Zlob trojan. `http://en.wikipedia.org/wiki/Zlob_trojan`, 2007.

[4] Alexa. Alexa the web information company. `http://www.alexa.com/`, 2007.

[5] M. Andrews. The dnssec lookaside validation (dlv) dns resource record, rfc 4431. `http://www.faqs.org/rfcs/rfc4431.html`, 2006.

[6] R. Arends. Dns security introduction and requirements, rfc 4033. `http://www.faqs.org/rfcs/rfc4033.html`, 2005.

[7] R. Arends. Protocol modifications for the dns security extensions, rfc 4035. `http://www.faqs.org/rfcs/rfc4035.html`, 2005.

[8] R. Arends. Resource records for the dns security extensions, rfc 4034. `http://www.faqs.org/rfcs/rfc4034.html`, 2005.

[9] D. Atkins and R. Austein. Threat analysis of the domain name system (DNS). `http://www.ietf.org/rfc/rfc3833.txt`, August 2004.

[10] S. Bellovin. Using the domain name system for system breakins. In *Proceedings of the Fifth USENIX UNIX Security Symposim*, June 1995.

[11] D. J. Bernstein. "dns software". `http://cr.yp.to/surveys/dns1.html`, 2002.

[12] M. Corpin. Rogue domain name system servers (reposted). `http://tinyurl.com/327b2k`, March 2007.

[13] B. Eckman and L. Zelster. An overview of the freevideo player trojan. `http://isc.sans.org/diary.html?storyid=1872`, 2006.

[14] J. Grizzard, V.Sharma, C. Nunnery, B. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study. In *Usenix Hotbots 2007*, April 2007.

[15] F. Hacquebord and C. Lu. Rogue domain name system servers part 2. `http://tinyurl.com/2sjgft`, September 2007.

[16] IANA. Special-use ipv4 addresses. http://www.faqs.org/rfcs/rfc3330.html, September 2002.
[17] J. Kristoff. Dns - open recursive name server probing. condor.depaul.edu/~jkristof/orns/, 2007.
[18] P. Mockapetris. Domain names - concepts and facilities. http://www.faqs.org/rfcs/rfc1034.html, 1987.
[19] P. Mockapetris. Domain names - implementation and specification. http://www.faqs.org/rfcs/rfc1035.html, 1987.
[20] J. Oberheide, M. Karir, and Z. Mao. Characterizing dark dns behavior. In *Fourth GI International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '07)*, 2007.
[21] E. Osterweil, D. Massey, and L. Zhang. Observations from dnssec deployment. In *The 3rd Workshop on Secure Network Protocols (NPSec)*, 2007.
[22] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang. Impact of configuration errors on dns robustness. In *Proceedings of ACM SIGCOMM 2004*, August 2004.
[23] N. Provos and P. Honeyman. ScanSSH - Scanning the Internet for SSH Servers. In *Proceedings of the 16th USENIX Systems Administration Conference*, December 2001.
[24] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The Ghost in the Browser: Analysis of Web-based Malware. In *Usenix Hotbots 2007*, April 2007.
[25] Sophos. Troj/qhosts-1. http://www.sophos.com/virusinfo/analyses/trojqhosts1.html, 2003.
[26] SpamHaus. Exploits block list. http://www.spamhaus.org/xbl/index.lasso, 2007.
[27] I. Symantec. Symantec gateway security products dns cache poisoning vulnerability. http://securityresponse.symantec.com/avcenter/security/Content/2004.06.21.html, 2004.
[28] Team Cymru. The team cymru bogon reference page. http://www.cymru.com/Bogons/, 2007.
[29] The Honeynet Project and Research Alliance. Know your enemy: Fast-flux service networks. http://www.honeynet.org/papers/ff/index.html, 2007.
[30] ThreatStop. Emergency public blocklist available. http://www.threatstop.com/, 2007.
[31] US-CERT. The continuing denial of service threat posed by dns recursion (v2.0). http://www.us-cert.gov/reading_room/DNS-recursion121605.pdf, 2006.
[32] P. Vixie. DNS complexity, April 2007.
[33] F. Weimer. Passive dns replication. In *17th Annual FIRST Conference on Computer Security Incident Handling (FIRST '05)*, 2005.
[34] D. Wessels. Dns survey: Cache poisoners. http://dns.measurement-factory.com/surveys/poisoners.html, 2007.
[35] D. Wessels. The measurement factory open recursive dns reports. http://dns.measurement-factory.com/surveys/openresolvers/ASN-reports/, 2007.
[36] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are ip addresses? In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'07)*, 2007.
[37] B. Zdrnja, N. Brownlee, and D. Wessels. Passive monitoring of dns anomalies. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2007.

APPENDIX

t	timeout
0	Return code 0: normal answer
1	Return code 1: format error
2	Return code 2: server failure
3	Return code 3: name error
4	Return code 4: not implemented
5	Return code 5: refused
TC	TC (Message truncated) bit set
RD	RD (Recursion desired) bit set
AA	AA (Is Authoritative) bit set
Z0	Z0 bit set
Z1	Z1 bit set
Z2	Z2 bit set
q	no queries listed in response
Q2	two queries listed in response
D	response included an answer record

Table 2. A list of designations used to fingerprint a DNS response packet.

Country	Forwarder	Open Recursives
Italy	82.53.187.212	316697
Italy	85.38.28.8	215087
Italy	85.38.28.5	178763
Netherlands	213.75.17.74	157619
Netherlands	213.75.17.76	157513
Netherlands	213.75.76.80	155518
Netherlands	213.75.76.79	155357
Italy	151.99.125.9	144516
Peru	200.48.225.130	123467
Italy	82.53.187.213	116104
USA	71.242.0.36	110472
USA	71.242.0.38	110463
USA	71.242.0.37	110163
Denmark	212.242.34.227	102616

Table 3. The table shows the top 10 recursive forwarding servers and the number of open recursive clients they serve.

Country	Number of forwarded-to resolvers	Percentage
USA	187990	28.5
Japan	58816	8.9
Germany	51554	7.8
Korea	28595	4.3
Brazil	26228	4.0
Taiwan	25886	3.9
China	24672	3.7
Russia	21620	3.3
Great Britain	21409	3.2
France	20819	3.2
Canada	16935	2.6
Poland	14654	2.2
Netherlands	13823	2.1
Italy	9369	1.4

(a) Location of all resolvers that are being used as forwards from open recursive resolvers.

Country	Number of forwarded-to resolvers	Percentage
China	231	20.1
Korea	187	16.3
USA	139	12.1
Japan	85	7.4
Poland	51	4.4
Germany	47	4.1
Spain	46	4.0
France	46	4.0
Turkey	38	3.3

(b) Location of resolvers that get forwards from more than $1,000$ different $/24$s.

Table 4. The table shows the location of forwarded-to resolvers and how many sub resolvers are forwarding to them.

Count	Percent	Query	Iquery	TC	Chaos
2694403	(26.3%)	0RDD	t	t	t
2325179	(22.7%)	0RDD	1q	1	0RDAAD
971579	(9.5%)	0RDD	5q	5q	t
722668	(7.0%)	0RDD	t	t	0RDAAD
403802	(3.9%)	0RDD	5q	5	0RDAAD
333807	(3.3%)	0RDD	t	2	t
291932	(2.8%)	0RDD	t	1	0RDAAD
239063	(2.3%)	0RDD	1q	t	0RDAAD
235423	(2.3%)	0RDD	1q	1	t
214298	(2.1%)	0D	t	t	t
177110	(1.7%)	0RDD	4q	0TCZ2	0RDAAD
175820	(1.7%)	0RDD	5q	5	t
117906	(1.1%)	0RDD	5q	t	t
105578	(1.0%)	0RDD	t	1	t
104396	(1.0%)	0RDD	t	5q	t

Table 5. DNS fingerprints of all open recursive resolvers.

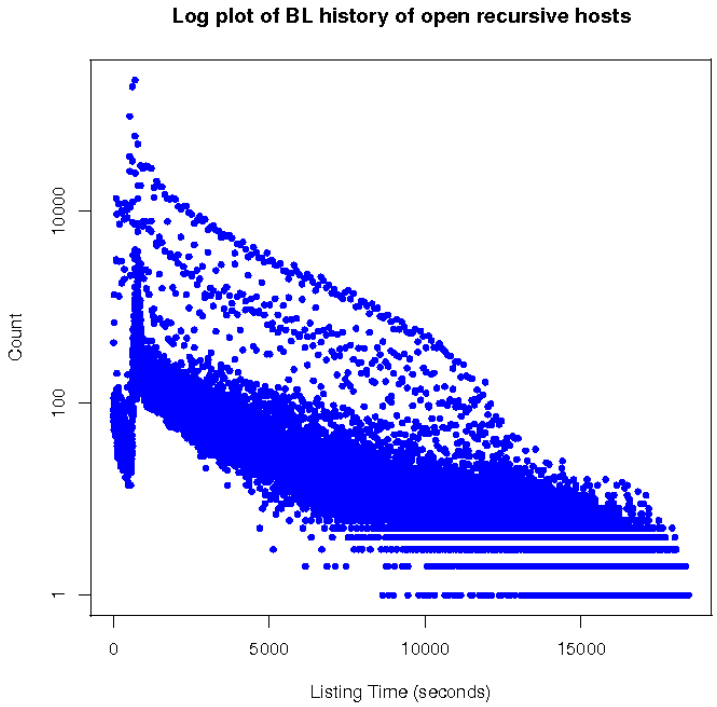


Figure 4. Logscale histogram of time open recursive hosts appeared on virus-related black lists, for 6-month period.

Count	Percent	Query	Iquery	TC	Chaos
63820	(39.5%)	0RDD	1q	1	0RDAAD
18852	(11.7%)	0RDD	t	t	t
16761	(10.4%)	0D	t	t	t
9773	(6.0%)	0RDD	4q	0TCZ2	0RDAAD
7715	(4.8%)	0RDD	t	1	0RDAAD
6655	(4.1%)	0RDD	1q	1	t
6589	(4.1%)	0RDD	1q	t	0RDAAD
3896	(2.4%)	0RDD	t	t	0RDAAD
3891	(2.4%)	0RDD	5q	5q	t
2359	(1.5%)	0RDD	t	1	t
2179	(1.3%)	0RDD	t	0TCZ2	0RDAAD
1358	(0.8%)	0RDD	1q	t	t
1306	(0.8%)	0RDD	4q	0TCZ2	t

(a) DNS fingerprints for resolvers overlapping with Google

Count	Percent	Query	Iquery	TC	Chaos
262003	(31.4%)	0RDD	1q	1	0RDAAD
228765	(27.4%)	0RDD	t	t	t
55892	(6.7%)	0RDD	t	t	0RDAAD
43647	(5.2%)	0RDD	t	1	0RDAAD
40598	(4.9%)	0RDD	5q	5q	t
32144	(3.8%)	0RDD	1q	1	t
26967	(3.2%)	0RDD	1q	t	0RDAAD
14844	(1.8%)	0RDD	t	2	t
12772	(1.5%)	0RDD	t	1	t
11366	(1.4%)	0RDD	5q	5	0RDAAD
7371	(0.9%)	0D	t	t	t
7339	(0.9%)	0RDD	1q	t	t
6439	(0.8%)	0RDD	5q	5q	0RDAAD
5461	(0.7%)	0RDD	0qD	t	t

(b) DNS fingerprints for resolvers overlapping with Storm/Peacomm

Table 6. DNS fingerprints for resolvers belonging to either Storm or Google.

Count	Percent	HTTP Server Version
8132640	(79.3%)	(no answer)
335827	(3.3%)	RomPager/4.07
205034	(2.0%)	Nucleus/4.3
175488	(1.7%)	Apache/1.3.37
161247	(1.6%)	(empty header)
148699	(1.4%)	Microsoft-IIS/6.0
142807	(1.4%)	(no header)
113518	(1.1%)	mini_httpd/1.19
69517	(0.7%)	GoAhead-Webs
59201	(0.6%)	Microsoft-IIS/5.0
55680	(0.5%)	RomPager/4.51
53925	(0.5%)	Apache
45083	(0.4%)	Apache/1.3.33
39643	(0.4%)	Apache/2.0.54
34710	(0.3%)	Apache/2.0.52
30806	(0.3%)	Apache/2.2.3
28567	(0.3%)	Apache/1.3.34

(a) HTTP Servers version for all open recursive resolvers

Count	Percent	HTTP Server Version
76371	(56.1%)	(no answer)
6175	(4.5%)	Microsoft-IIS/6.0
4273	(3.1%)	Apache/1.3.33
3764	(2.8%)	Microsoft-IIS/5.0
3351	(2.5%)	Apache
3211	(2.4%)	Apache/2.2.3
3204	(2.4%)	Apache/2.0.54
3122	(2.3%)	Apache/1.3.37
2478	(1.8%)	Apache/2.0.52
2363	(1.7%)	(no header)
1944	(1.4%)	RomPager/4.07
1727	(1.3%)	Apache/1.3.34
1723	(1.3%)	Apache/1.3.27
1554	(1.1%)	Apache/2.2.4
1229	(0.9%)	Nucleus/4.3
1138	(0.8%)	Apache/2.0.40
964	(0.7%)	Apache/2.0.55
912	(0.7%)	Apache/2.0.59
907	(0.7%)	Apache/2.0.46
880	(0.6%)	Apache/1.3.26
854	(0.6%)	Apache/2.0.53

(b) Open recursive resolvers overlapping with Google

Count	Percent	HTTP Server Version
534368	(64.0%)	(no answer)
92201	(11.0%)	RomPager/4.07
60484	(7.2%)	Nucleus/4.3
55624	(6.7%)	mini_httpd/1.19
33938	(4.1%)	(empty header)
19679	(2.4%)	RomPager/4.51
14453	(1.7%)	(no header)
5170	(0.6%)	Unknown/0.0
3915	(0.5%)	GoAhead-Webs
1640	(0.2%)	Microsoft-IIS/6.0
1300	(0.2%)	nginx/0.5.17
1136	(0.1%)	httpd
966	(0.1%)	Apache/0.6.5
843	(0.1%)	ZyXEL-RomPager/3.02
748	(0.1%)	Apache/1.3.33
723	(0.1%)	Apache/2.2.3
653	(0.1%)	Apache/1.3.27
503	(0.1%)	Apache/2.0.54
475	(0.1%)	Microsoft-IIS/5.0
464	(0.1%)	Apache
458	(0.1%)	Apache/2.2.4

(c) Open recursive resolvers overlapping with Storm

Table 7. HTTP Servers version of open recursive resolvers.