Pollution Resilience for DNS Resolvers

Andrew Kalafut
Computer Science Dept
Indiana University
Bloomington, IN
Email: akalafut@cs.indiana.edu

Minaxi Gupta
Computer Science Dept
Indiana University
Bloomington, IN
Email: minaxi@cs.indiana.edu

***Abstract*—The DNS is a cornerstone of the Internet. Unfortunately, no matter how securely an organization provisions and guards its own DNS infrastructure, it is at the mercy of others' provisioning when it comes to resolutions its resolvers perform on behalf of its clients – even one compromised DNS server in the Internet can mislead an organization's clients to fake look-alike phishing Web sites or malware-serving sites, among other things. In this paper, we propose a self-defense mechanism where the DNS resolvers collect a small amount of additional information for the DNS responses they receive and maintain a history of previous responses to guard their clients against misleading information from compromised DNS servers in the Internet. Any organization can choose to enhance its resolvers with our mechanism unilaterally, unlike DNSSEC, which can ensure correctness of information only if the remote DNS server deploys it.**

I. Introduction

The Domain Name System (DNS) maps host names into the IP addresses. Nearly all applications with any Internet functionality make use of it. There are several steps an organization can take to provision its DNS servers securely. While those steps ensure the availability and correctness of records belonging to the organization to the rest of the world, they do nothing to guard its clients from being misled when they visit remote sites, since the DNS resolver for its clients has to trust the information provided by remote DNS servers. Specifically, an attacker who manages to compromise a remote DNS server can modify records to mislead the clients to fake look-alike sites used for phishing, or simply to sites serving malware. With a recent study noting that 68% of the organizations surveyed had suffered malware-related attacks on their DNS servers (worms, viruses, Trojans etc.) [1], the worry of being misled by compromised DNS servers is a real threat, not just paranoia.

In this paper, we propose a mechanism to defend the resolvers (and hence, the clients) against incorrect responses from compromised DNS servers in the Internet. In developing our mechanism, we leverage two observations we make from real-world data sets: 1) *Host name to IP address mappings stay constant 96-99.4% of the time* and 2) *Even when they change, the mappings stay within the same autonomous system (AS) 97-99.8% of the time*. These observations imply that any changes to a different AS should be scrutinized further. Consequently, our mechanism requires the resolvers to collect a small amount of additional information for the DNS responses they receive and maintain a history of previous responses to guard their clients against misleading information from compromised DNS servers in the Internet. Any organization can choose to enhance its resolvers with our mechanism unilaterally, and receive the benefits, irrespective of who else is deploying it. In addition to protecting against compromised authoritative servers, our proposed enhancement can also help protect the resolvers against cache poisoning attempts. Although cache poisoning vulnerabilities have been known for many years, problems have still been found and patched in very recent DNS server software [2]. Kaminsky recently identified new vulnerabilities not specific to a particular implementation [3]. Thus, mechanisms to guard against cache poisoning are essential.

Our mechanism incurs very little latency for over 99% of DNS resolutions. While it does require maintaining history, in the majority of cases, this overhead does not affect the client latency because the resolver can gather this information out-of-band, say when it is experiencing a low load. For the remaining, the expected overhead is about half a second, which is significant compared to the median DNS resolution time of 164ms, but is incurred only rarely.

II. Current Resolver Operation

Logically speaking, each organization has two types of DNS servers. The first, internal DNS servers, typically referred to as *resolvers*, are only for use by clients internal to the organization. The resolvers query remote DNS servers on behalf of the clients and maintain a cache of the received responses. The second, external DNS servers, typically referred to as *authoritative DNS servers*, only respond to queries from remote resolvers for hosts belonging to the organization. Each domain in this tree must have a set of authoritative DNS servers which answer DNS requests about the domain, and give the location of authoritative DNS servers of its sub-domains. For security, it is recommended to separate the resolver and authoritative DNS servers.

When a resolver receives a client query to resolve a host name to an IP address, it follows a series of steps to respond to the request. First, it consults its local cache. If it finds an appropriate record for that information that is valid, it returns it to the client. If a valid copy of the appropriate record is not found, then the resolver traverses the DNS hierarchy to satisfy the client request.

As an example, consider the case when the resolver has to consult the DNS hierarchy to find the IP address for `www.example.com`. The resolver first looks for authoritative DNS servers for `example.com` in its cache. If any are found, they are queried for the IP address. Otherwise, the resolver checks if it has cached the authoritative servers for `.com`. Assuming it does have the information about `.com`, it asks the authoritative servers for `.com` for the authoritative servers for `example.com`. Upon receiving these entries, the resolver contacts one of them to get the entry for `www.example.com`. In the worst case, the resolver must go to the *root* DNS servers, fetch authoritative DNS server entries from them for `.com`, use one of those entries to fetch the authoritative servers for `example.com` and then get the entry the client desires.

In addition to translating names to IP addresses, the resolvers are used for other functions, such as looking up email servers or servers providing other services. The process of resolving these queries is similar to the process of resolving the host name to IP mapping. Aside from the type of records being requested, the main difference in the look-up process for these is that the records returned contain a host name of the server providing the service, instead of an IP address. Therefore, once the host name of the desired server is found, an additional step will be required to find its IP address. Subsequently, we focus mainly on ensuring the correctness of DNS responses containing host name to IP mapping. However, the system we describe can easily work for most other record types with only one extra step of translating the host name returned to an IP address.

III. Our Approach

Our approach to defend the resolvers against incorrect responses from compromised authoritative DNS servers in the Internet is based on the following observations we make from the DNS resolutions at our department.

- The host name to IP address mappings stay constant for 96.0-99.4% of the hosts in the Internet.
- Even when they change, the new mappings often stay within the original autonomous system number (ASN). 96.9-99.8% host name to ASN mappings stay constant.

We leverage these observations to equip the local resolvers with additional information about the DNS resolutions they perform for their clients and with historical information about previous resolutions. This information can then be used to flag receipt of potentially incorrect resolutions.

A. Threat Model and Scope

Our threat model implicitly assumes that concerted efforts to compromise DNS hierarchy are in general not possible. For example, we assume that an attacker can compromise an organization's authoritative DNS but would not be able to synchronize that compromise with 1) the compromise of `.com` server(s), or 2) a DoS attack on the `.com` or root DNS servers, or 3) an illegal edit to the *whois* entry. While the possibility that a highly determined adversary will be able to orchestrate various compromises cannot be ruled out, each of these additional compromises together would be very hard in practice.

Our mechanism focuses on protecting the resolvers against changes to DNS records at compromised authoritative servers in the Internet. Several other DNS-related threats exist which we do not attempt to cover. First, the clients may be tricked into visiting the wrong DNS resolver instead of one belonging to their organization, for example by malware which changes this setting [4]. A combination of anti-virus software and spam-filtering programs can avoid this situation. Second, DoS attacks on any part of the DNS hierarchy could prohibit resolvers from receiving responses in the first place. Ballani and Francis's proposal to use stale cache entries in such cases [5] offers a solution to this problem and could easily be incorporated into our proposal, as both require maintaining similar types of information. Finally, we only protect against DNS compromises. Protecting against compromises to the server being visited is outside the scope of our proposal.

B. Constancy of DNS Resolutions

We began by gathering four one-week snapshots of all DNS queries and responses seen at our department's internal DNS server. These are all the queries from hosts on our network and the responses to them. All of the snapshots are one week in length. Snapshots 1 and 2 are directly adjacent weeks, as are 3 and 4. However, there is a gap of nearly a month between 2 and 3. Details of these snapshots are in Table I.

TABLE I
An overview of four snapshots of DNS queries

	1	2	3	4
Start Date	2008-07-01	2008-07-08	2008-08-07	2008-08-14
Duration	7 days	7 days	7 days	7 days
Queries	20,040,357	37,216,219	29,270,728	33,870,313
Queries for IP	2,422,334	2,348,936	5,490,942	8,569,120
Host names	60,913	60,393	113,311	101,311

We compare adjacent snapshots for host commonality. We find that over 25% of the host names are common across both snapshots. For common host names, we look at how the IP addresses for them have changed between the two snapshots. Whenever all the IP addresses for a host name change, we use BGP routing tables from the first day of the snapshot to translate the IP addresses to AS numbers (ASNs), and check if all the ASNs for the host names have also changed. Table II shows the results of this analysis. We find that most host names have common IP addresses in adjacent snapshots. In most cases, all the IP addresses remain the same. Only 0.6-4.0% hosts have no common IP addresses. In many cases when the IP addresses change, the ASNs do not. In only 0.2-3.1% cases do the host names have no common ASNs. *We conclude that most hosts stay in the same ASN over time but the popular sites are slightly more likely to change IP addresses and ASNs.*

C. Enhancements to Resolver Operation

Our proposal requires three changes to resolver functionality. First, it requires that the resolvers gather additional infor-

TABLE II
COMMONALITY ACROSS ADJACENT DNS SNAPSHOTS

	1 and 2	2 and 3	3 and 4
Host names in both snapshots	15,470	16,950	27,122
No common IP	180 (1.2%)	683 (4.0%)	162 (0.6%)
No common ASN	126 (0.8%)	521 (3.1%)	66 (0.2%)

mation for every DNS resolution they receive and maintain a limited history of it. Second, it requires that the resolvers check new responses against this history to see if their information has changed. Third, it requires resolvers to conduct additional heuristic checks before passing the responses they receive to the clients, if the history check did not indicate a match.

1) Maintaining history: Currently, resolvers maintain in their cache the latest IP address(es) and time to live (TTL) corresponding to each host name. They also maintain the name, IP address, and TTLs for the corresponding authoritative DNS server(s) that could be used to fetch this information. We require a few key additions to these pieces of information. First, we require that the resolvers also maintain the ASN corresponding to each IP address, including that of the authoritative DNS server(s). A relatively low overhead way to retrieve ASN for any IP address is to perform a *longest prefix match* of that IP in the BGP routing announcements, which contain ASN information. A periodic download of BGP announcements is sufficient toward this goal and no real-time fetching of information is required since ASNs change relatively infrequently. Second, we require that the resolvers fetch the *whois* entries corresponding to each ASN. The *whois* protocol originally was designed for looking up information about registered domains, but now supports look-up of other identifiers, such as ASNs. (We discuss overheads associated with fetching these pieces of information in Section III-E.) Lastly, we require that the resolvers maintain a history of previous resolutions, storing all this information beyond the length of time specified in the TTL, when it would currently be erased.

2) Using history to flag potentially incorrect responses: The very first check a resolver conducts in our scheme upon receiving a host name to IP address mapping is to see if there is a commonality in the IP addresses received with respect to the history. If there is, nothing else needs to be done except updating the history information with the new count of the times the same IP addresses were received. In this case, the response can be sent back to the client. If all of the IP addresses change, we require the resolver to retrieve the ASNs corresponding to the new IP addresses using a locally cached copy of the BGP routing tables and check if the ASNs of the new IP addresses are the same as the the ASNs of the old IP addresses. If it is, the server has changed IP addresses within its ASNs. In this case also, nothing else needs to be done and the response can simply be sent back to the client and history updated with the number of times the new IP address was seen.

If all of the ASNs corresponding to the new IP addresses differ from the ASNs corresponding to the old IP addresses, the resolver needs to conduct additional checks to confirm that the new information is a valid change in the remote domain's configuration and not a compromised remote authoritative DNS server's attempt to mislead. Based on the observations we made in Section III-B, we require that the resolvers conduct two checks, listed below:

1) Correct authoritative DNS server: The very first check the resolver conducts is to check if the authoritative DNS server information contained in the response or the IP address of the sender of the response matches any of the authoritative DNS server information contained in history. If there is a match, it implies that the DNS server we received a response from is a valid one. If there is no match, we require the resolver to go up one level in the DNS hierarchy and explicitly make a DNS query to fetch information about the authoritative DNS servers and query them in turn. If the responses from the authoritative DNS server records so fetched do not match what was received in the original response, the resolver knows the original response is bad and should be discarded. However if they do match, the resolver knows it got its response from a valid authoritative server, but can not be sure the server itself was not compromised, and must move on to the next check.

2) ASN-based *whois*: Several large organizations in the Internet maintain more than one ASN. We use *whois* entries for ASNs to determine if a change between ASNs is really just a change between ASNs belonging to the same organization. It is important that the resolvers account for all such changes and not flag them as attempts to mislead their clients. The *whois* information we use is not the typical *whois* entries for domain names, instead it is the *whois* information containing information about the owner of each ASN. While there is concern about *whois* data being untrustworthy, our preliminary investigation has shown that *whois* entries for ASNs appear more reliable than that for domains, and the entries appear to be in a consistent format. Additionally, ICANN has recently began efforts to study, audit, and improve the accuracy of *whois* [6].

Unlike what is often found in *whois* entries for domain names, in *whois* entries for ASNs there is a consistent format, allowing an automated comparison of the information contained in these records. The ASN *whois* records we investigated all contained sufficient information to enable a comparison of a combination of city, state, country, zip code, phone area code, and email address (especially, the domain name portion of the email address) across multiple ASNs and infer if they belonged to the same domain. We compare this information between the ASNs contained in the stored history, and the ASNs of the addresses in the response we received. If it matches, this indicates the ASN change was within the same organization, so we consider the response to be valid. If it does not match, then we consider that the records were likely maliciously modified by a server compromise or some other method in order to point somewhere controlled by the attacker, so the response is not considered valid.

D. Handling flagged DNS responses

There are several options for what a resolver should do when it receives records it believes are incorrect. One option is to return a DNS error to the application, denying access to the new records. This is harmful in the case of false positives and therefore not a good option. Another option is to return the old records from the history, even though they may be outdated, however this could lead to problems if the records are no longer valid. A third option is to report an error at the user level and let the user decide what action to take. However, it may be too much to ask users without technical knowledge. Perhaps the best option would be to enable communication between this system and anti-spam systems and browser toolbars, which could incorporate information from this system in making their decisions.

E. Overheads of Our Approach

In order to maintain the history we need to do our checks, we add a few steps to all DNS resolutions. For each IP address not found in history, there is a translation the corresponding ASNs. Also, for each ASN not contained in history, there is a a *whois* look-up. Since 1/4th of our host names are common between consecutive snapshots, these steps concern the rest of the look-ups and can be done in batches after the DNS results have been returned to the user. Thus, they add nothing to the user perceived latency. We have at times in the course of our research added an additional 150% of its normal load to our departments DNS server, which it has been able to handle without any noticeable degradation in performance. Assuming that a *whois* look-up has a similar processing overhead as a DNS look-up, servers provisioned similar to ours with respect to their load should be able to handle our system.

Of the 1/4 host names that are common across snapshots, we find that only in 0.6-4.0% of these cases do the IP addresses change entirely. In each of these cases, a user-visible overhead will be added of doing the IP to ASN mapping. We have found this mapping to take a median of $1.2\mu s$ on a 3.2GHz Pentium IV machine with 2GB of RAM. This time is negligible in comparison to the median 164ms for a DNS resolution measured on our department network. In the cases where the ASN match also fails (0.2-3.1%), then other checks will be necessary. These checks can be done in parallel, meaning that the overhead will be that of the longest check. The median time we measure for a *whois* look-up is 567ms, just over half a second. This is significant overhead, but these checks will only need to be done a fraction of the time.

IV. Discussion and Open Issues

Various aspects of the proposed system need further consideration and a detailed investigation. We discuss them here.

False positives and negatives: False positives and false negatives are both possible in our approach, although it is difficult to determine how often they would actually occur since we do not know how often the compromises we are trying to defend against actually take place.

False negatives (not marking a DNS response as bad when it should be) are possible only when attacks are well coordinated across several systems. Specifically, the *whois* information for the ASNs of the attacker's IP addresses would have to be updated to match that of the organization being attacked. In general, this would not be possible. Thus, we believe that our system would not have any false negatives.

False positives (marking a DNS response as bad when it should not be) are somewhat more of a concern. If a Web site were to legitimately move and have its information change in all the systems we use for our checks, this would cause a false positive. One scenario where this could occur is a change in hosting providers. Under this system, a Web site would change its IP addresses and ASN to that of the new hosting provider. The *whois* and authoritative DNS server information would also correspond to that of the hosting provider, and the host at the old hosting provider may stay up to be used for other purposes. It is difficult to determine how much of a concern this would be in practice. This problem could be prevented by a gradual change in providers, where records for both coexisted for some time. The new ones would then be added to the history before the old ones disappeared.

Bootstrapping: When we encounter a new host, we have no history information for them. This forces us to simply accept the information they give us without question. If when we first see them, their DNS servers are already compromised, we can not provide protection. This can be partly solved by sharing information between DNS servers, although that may open up additional trust issues.

How much history should be maintained? While the history we are keeping is only a small amount of information for each domain, storing too much history is still undesirable. We compare our snapshots with another week-long snapshot from before, beginning on May 14, 2008. We find that the overlap in terms of host names is still high as when we compare adjacent snapshots, comparing with our snapshot 4, 20212 host names are common. However, when we look at IP addresses and ASNs, of these host names 7.4% have changed IP addresses and 4.7% have changed ASNs. This is significantly worse than the comparisons of snapshots we have from directly adjacent weeks. Between snapshots 1 and 2 we saw 1.2% change IP addresses and 0.8% change ASNs. Results were even better between weeks 3 and 4. This indicates that while recent history does good, older history as indicated by the three month gap we see here or even the one month gap between snapshots 2 and 3 is less useful. While older history information should be retained if no newer history information for a given host name is available, when new history is available, it is likely best not to keep older information beyond a couple weeks, and certainly not for months.

Defending other DNS functionality: While we focused on the common case of host name to IP address mapping, DNS has other functionality. Most other common record types an attacker can take advantage of, such as aliases and mail servers, resolve one host name to another. Our system naturally extends to protecting these. By keeping this history of these mappings, and history of the address mappings for the target host name, our system naturally extends to protecting these records. For example to check if a new mail server record is accurate, the DNS resolver would find the mapping from domain name to mail server host name, and the mapping from the mail server host name to its IP address. Both of these pieces of information would be stored in history from previous look-ups. If the host name of the mail server did not match, the DNS server would resolve the address associated with it and check if these match or their ASNs match, and apply the two heuristics if necessary.

V. Related Work

There are several other steps a security-conscious organization should take to provision its DNS servers well. It should separate its internal DNS servers, resolvers used by its own clients, from its external DNS servers, the authoritative servers that the rest of the Internet contacts [7], in order to prevent their use as *open resolvers*, which can be misused in myriads of attacks. It should also allow for sufficient redundancy in its authoritative DNS servers, both in number and geographical diversity, to enhance their availability in the face of denial-of-service (DoS) attacks [8], [9]. Further, deploying DNSSEC (DNS Security Extensions) [10] will help ensure that the authenticity of the DNS records being served can be verified by others. While all these steps can go a long way toward making the DNS infrastructure safer as a whole, these practices do not seem to be well adopted [11], [12]. Further, they do very little to enable an organization to protect its own clients from attacks arising out of security lapses in the provisioning of DNS servers in the rest of the Internet.

History information in DNS has been used for different purposes in other work. Ballani and Francis [5] proposed using history information to defend against Denial of Service (DoS) attacks on the DNS. They added a component called the *stale cache*, which would store DNS records after they are expired. The resolver then falls back to using these stale entries if the authoritative DNS servers for a requested name were unavailable. This idea could be integrated into our scheme, adding DoS attack protection. Cohen and Kaplan [13] propose *simultaneous validation* to improve performance. Under this system, expired DNS records will be returned to the user programs, and simultaneously the DNS resolver will send a request for new records. The resolver communicates with the user program to notify it if the new records ended up not matching the expired ones. This work uses the history for performance, not security, and so does not attempt to validate the new records as we do.

Poole and Pai [14] propose ConfiDNS, which is a peer-to-peer system to improve DNS security. While the main focus of their work is on agreement between peers, they also augment this with history information. If not enough peers confirm a DNS record in their system, they are still willing to believe the record if the IP address has remained unchanged for a certain amount of time, which depends on the number of peers they have agreeing. The history portion of this work could also benefit from our system. While their history focuses just on IP addresses, ours includes AS information which they could take advantage of.

VI. Conclusion

While most of the DNS security efforts focus on improving the DNS infrastructure as a whole, our proposal focuses on self-defense. The checks we use are heuristic in nature, and situations are likely to occur where they are not accurate. However, it is still beneficial in helping allow an organization means to protect its own DNS resolutions. Further, it can be deployed unilaterally by any organization irrespective of who else in the Internet is deploying it.

Firewalls offer the first line of defense against malware. Anti-virus software and intrusion detection systems add to that defense. Spam filtering not only filters spam but also emails containing phishing scams and malware attachments. Web browsers and browser toolbars add to these shields by warning the users against visiting phishing sites. Our approach can complement each of these other efforts by potentially preventing users from visiting fake look-alike servers and hosts in the Internet. A spam filter or browser toolbar could potentially be improved by incorporating information provided by this system.

Acknowledgments

We would like to thank Rob Henderson for providing anonymized DNS traffic from our department's DNS server.

References

[1] M. Cooney, "Is IT losing the battle against DNS attacks," Network World, Jul. 2007.

[2] A. Klein, "BIND 9 DNS cache poisoning," Whitepaper, 2007, http://www.trusteer.com/docs/bind9dns.html.

[3] D. Kaminsky, "Black ops 2008: It's the end of the cache as we know it," Black Hat USA 2008 presentation, Aug. 2008.

[4] Symantec, "Trojan.Flush.K," Symantec Security Response, http://www.symantec.com/security_response/writeup.jsp?docid=2007-011811-1222-99&tabid=1.

[5] H. Ballani and P. Francis, "A simple approach to DNS DoS mitigation," in *ACM SIGCOMM Hot Topics in Networking (HOTNETS)*, 2006.

[6] Internet Corporation for Assigned Names and Numbers (ICANN), "Update: ICANN projects underway to improve whois accuracy," http://www.icann.org/announcements/announcement-2-21dec07.htm.

[7] A. Householder, B. King, and K. Silva, "Securing an internet name server," CERT Coordination Center Whitepaper, 2002.

[8] D. Barr, "Common DNS operational and configuration errors," IETF RFC 1912, Feb. 1996.

[9] R. Elz, R. Bush, S. Bradner, and M. Patton, "Selection and operation of secondary DNS servers," IETF RFC 2182, Jul. 1997.

[10] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," IETF RFC 4033, Mar. 2005.

[11] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang, "Impact of configuration errors on DNS robustness," *ACM SIGCOMM Computer Communications Review (CCR)*, vol. 34, no. 4, pp. 319–330, 2004.

[12] A. Kalafut, C. Shue, and M. Gupta, "Understanding implications of DNS zone provisioning," in *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2008.

[13] E. Cohen and H. Kaplan, "Proactive caching of DNS records: Addressing a performance bottleneck," in *Symposium on Applications and the Internet*, 2001.

[14] L. Poole and V. Pai, "ConfiDNS: Leveraging scale and history to improve DNS security," in *USENIX Workshop on Real, Large Distributed Systems*, 2006.