Hold-On: Protecting Against On-Path DNS Poisoning

Haixin Duan*, Nicholas Weaver†¶, Zongxu Zhao*, Meng Hu*, Jinjin Liang*, Jian Jiang*, Kang Li‡ and Vern Paxson†§
* Tsinghua University, Beijing, CN
duanhx@tsinghua.edu.cn
† International Computer Science Institute, Berkeley, CA, USA
‡ University of Georgia, Athens, GA, USA
§ University of California, Berkeley, CA, USA
¶ University of California San Diego, CA, USA

***Abstract*—Several attacks on DNS inject forged DNS replies without suppressing the legitimate replies. Current implementations of DNS resolvers are vulnerable to accepting the injected replies if the attacker's reply arrives before the legitimate one. In the case of regular DNS, this behavior allows an attacker to corrupt a victim's interpretation of a name; for DNSSEC-protected names, it enables denial-of-service.**

We argue that the resolver should wait after receiving an initial reply for a "Hold-On" period to allow a subsequent legitimate reply to also arrive. We evaluate the feasibility of such an approach and discuss our implementation of a prototype stub resolver/forwarder that validates DNS replies using Hold-On. By validating the IP TTL and the timing of the replies, we show that the resolver can identify DNS packets injected by a nation-state censorship system, and that it functions without perceptible performance decrease for undisrupted lookups.

I. Introduction

The Domain Name System (DNS) provides a critical network service, and faces a variety of attacks ranging from blind packet injection to active man-in-the-middle attacks. One attack of concern regards DNS poisoning based on packet injection, where an attacker who can observe and inject traffic inserts fake replies to queries. Several types of adversaries can employ such attacks, including attackers using systems on shared WiFi networks, ISPs seeking to impose content-based usage polices, and government censorship [1].

One particular design choice of DNS makes these attacks easy. The DNS standard recommends that a DNS resolver returns an answer as soon as it receives a matching reply [2], in order to provide a reply as quickly as possible. In addition, even DNSSEC-validating resolvers likely will suffer a denial-of-service attack upon receipt of an injected reply: the non-validating response leads the resolver to return a response of "Bogus" [3] unless it continues to wait for a reply that properly validates.

We explore the opportunity of countering DNS injection attacks based on the observation that packet injection (rather than full man-in-the-middle attacks) cannot suppress the receipt of legitimate replies. Thus, if resolver receives a reply sooner than expected, instead of returning the result immediately, it can wait for a "Hold-On" interval to see whether additional responses arrive.

The key questions for this approach are (1) to what degree such ambiguous replies occur in normal traffic, which will lead to Hold-On introducing different resolver behavior than occurs today, and (2) how much extra delay users encounter due to the use of Hold-On. Our evaluation shows that receiving two differing replies to the same question occurs only very rarely in normal traffic, which establishes that this condition allows for effective anomaly detection. We also present preliminary results suggesting that the extra delay imposed on users is quite minor. We have implemented a DNS proxy that uses Hold-On and evaluate its effectiveness against a widely deployed network censorship tool. We find that our prototype can effectively filter out fake DNS replies, and does not appear to introduce any perceptible increase in delay.

II. Overview of the Problem Space

A. Taxonomy of attacks

Attackers against DNS fall into three categories: **off-path**, **on-path**, and **in-path**.

An off-path adversary lacks the ability to observe DNS queries and responses. Such an attacker will generally employ some means to trigger specific DNS lookups, but must guess the transaction ID [4], [5] and any other entropy (such as the source port and 0x20 encoding [6]) in the request to forge a reply that the resolver will accept. Off-path adversaries generally generate numerous packets in hopes of matching the request. Additionally, because resolvers do not issue new queries for a name that is already cached, off-path adversaries have difficulty targeting stub resolvers, since stubs, unlike recursive resolvers, do not generally accept and promote glue entries (the behavior leveraged by [5]).

An on-path adversary has the ability to passively observe the actual lookups requested by a resolver. On-path adversaries can directly forge DNS replies that match the full set of criteria used by the resolver to validate answers (other than use of DNSSEC). As long as a forged reply arrives at the resolver before the legitimate one, the resolver will accept the injected answer and become poisoned.

Absent a denial-of-service attack on legitimate servers, both off-path and on-path adversaries lack the ability to suppress legitimate responses. Thus, both of these adversaries necessarily

create an observable artifact: the victim, if it waits sufficiently long, will receive both the attacker's packet and the legitimate reply. (We employed a similar form of this anomaly to detect TCP reset injection attacks [7].) Only an *in-path* adversary, capable of blocking and modifying packets, can prevent the legitimate reply from reaching the victim.

Although in-path approaches have more power, on-path approaches have several advantages, making their use appealing for attackers. Censorship tools commonly use on-path rather than in-path techniques to ease deployment and to make the system failure and load tolerant, as the censorship system can then operate on a traffic mirror rather than the live traffic.[1] Similarly, on-path WiFi packet injection works without modifying drivers, but suppressing legitimate replies requires hardware-specific access to the low-level air interface to detect and squelch a broadcast in flight.

B. Vulnerability of current implementations

Systems that implement the DNS standard [2], [8] are vulnerable to on-path spoofing, despite the presence of the later legitimate reply, because the resolver attempts to "get the answer as quickly as possible" [2]. Upon receiving a reply, the resolver checks the ID field in the header and then will "verify that the question section corresponds to the information currently desired" [8]. Clearly, these steps do not provide sufficient diligence, as the design goal of quickly returning an answer causes the resolver to return the attacker's value.

DNSSEC adds cryptographic authentication to prevent the acceptance of invalid DNS replies [9], [10], [3]. Although attackers cannot redirect victims using spoofed replies, they can still perform denial-of-service attack, which will often suffice to satisfy a censor's objective. DOS occurs because the resolver will attempt to process the attacker's packet, determine that the DNSSEC signature is absent or invalid, and immediately return "Bogus", depriving the client from the ability to connect to the host corresponding to the name. Because of this behavior, DNSSEC does not suffice as a replacement for a mechanism such as Hold-On: resolvers need to maintain an open port for a period of time in order to attempt to validate all responses received for a query, not just the first.

C. Other related work

DNS has a long history of poisoning attacks [4], [5], [11], [12]. Beside those mentioned above, several previous efforts counter DNS poisoning attack by increasing the difficulty of blindly injecting DNS answers [13], [14], [6], [15]. These efforts focus on deterring off-path injection by increasing the information entropy required to match a valid DNS reply. Our work, however, addresses the threat from attackers that can observe queries, which allows them to circumvent these previous defenses.

[1]TCP traffic control tools also have used this vantage point. For example, Comcast deployed Sandvine's Policy Traffic Switch devices to disrupt BitTorrent traffic in an on-path configuration [7], even though the devices themselves supported in-path operation.

Poisoning attacks based on on-path injection are not limited to DNS. Malicious injection, such as TCP RST and ICMP unreachable messages, have been used in both individual attacks [7] and ISP-scale censorship [16], [17]. Similar to DNS poisoning, traffic sent from the remote peer of the legitimate communication will still arrive at the victim after these malicious injections. Therefore, the use of Hold-On mechanisms similar to those explored here will likely have applicability to deter these malicious injections as well.

III. Hold-On and Diligent Validation

As a consequence of the inability for on-path attackers to suppress legitimate replies, we investigate the benefits of stub resolvers or forwarders waiting for a "Hold-On" period to allow subsequent legitimate replies to arrive. Part of this procedure entails validating replies with more diligence when a resolver receives two or more replies for the same question. This improvement effectively protects against DNS injections in the case of non-disruptive attacks, where the attacker lacks the ability to to block either the resolver's request or the authority's response.

A. Assumptions

We predicate our approach on the following assumptions, which we view as reasonable based on our experience with censorship activity that employs on-path injectors:

(1) The user under attack or censorship is able to access a trustworthy recursive resolver outside of the attacked or censored networks, such as Google Public DNS [18] and OpenDNS [19], which they frequently use. In particular, in the censorship case, we assume that the censor does not block access to this resolver, which we argue is a plausible assumption given the large number (158,364 in January 2012) of known open resolvers [20].

(2) The attacker/censor injects fake responses according to a blacklist rather than a whitelist. That is, the user knows some non-sensitive domain names that can be used to measure normal (non-interfered by the attacker) communication between the client (stub resolver) and the DNS server (recursive resolver).

(3) The attacker injects fake replies as quickly as possible in order to ensure that the replies arrive earlier than the legitimate ones. Hence, the injection mechanism will transmit immediately upon seeing the client's request. The mechanism cannot wait for the arrival of the legitimate reply from the server because by doing so, the injection may arrive after it, and fail to work.

(4) The attacker cannot construct a properly signed DNSSEC response.

Based on these assumptions, the stub resolver can estimate when it expects legitimate replies to arrive, in order to discern between injected replies and correct ones.

B. Hold-On and Validation

The stub resolver or forwarder needs to first learn the expected RTT and hop-count distance (in terms of expected

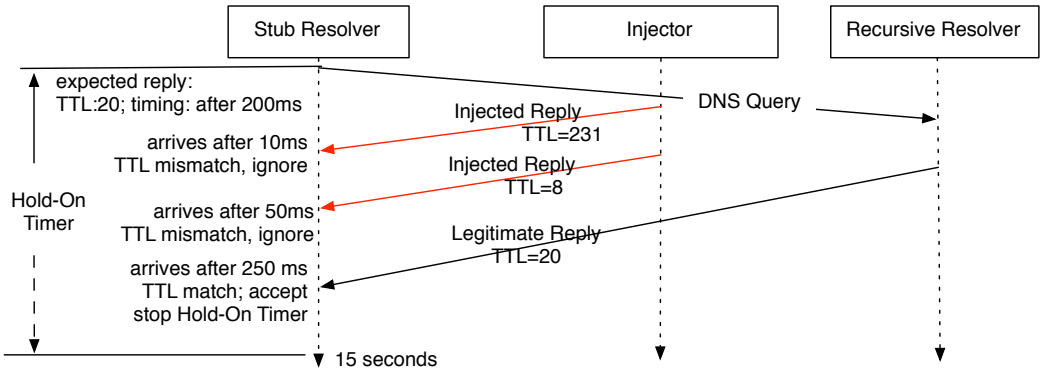

Fig. 1. Hold-On while waiting for a legitimate DNS reply.

TTL) associated with communication involving its remote recursive resolver, which it does using active measurement. (Recall that we presume the remote resolver lies outside of the censored network.) Upon start-up, the resolver issues a series of non-sensitive queries to measure the initial RTT and TTL seen on arriving replies for entries cached at the remote resolver by repeatedly querying for the same name. During this period, the resolver maintains an open port for an additional period to validate that an on-path adversary has not tampered with these initial measurements by injecting replies. During normal operation, the stub resolver also continually updates these values based on passive measurements of its ongoing traffic.

Given estimates of the legitimate RTT and TTL, the resolver works as shown in Figure 1:

(1) After issuing a DNS query, the resolver starts its Hold-On timer. A natural setting for the timer would be 15 seconds, as this reflects the default timeout value for both the BIND resolver [21, p. 108] and Microsoft Windows [22]. Naturally, in most cases the resolver will return much sooner, unless the remote resolver is unreachable.

(2) When the resolver expects a DNSSEC-protected response, for each reply it performs a local signature validation. It returns to the client the first fully validated reply. If it finds all replies as either Insecure, Bogus, or Indeterminate [3, p. 20], and the Hold-On timer expires, the resolver returns a DNSSEC error.

(3) Without DNSSEC, upon receiving a reply before the Hold-On timer expires, the resolver performs two additional validations:

- **Timing**. Does the reply arrives too early? The test we use here is for replies that arrive sooner than half of the expected (measurement-derived) RTT. We note that the resolver could also determine this threshold more precisely by measuring known injections in the resolver's actual environment by generating queries for censored names to non-existent resolvers.
- **TTL**. Does the TTL field in the IP header have the expected value(s)? We assume that the route between the remote DNS server and the client is stable in at least short periods (such as 5 minutes), so we can get and update the expected TTLs by periodical measurement.

Upon observing either of the above mismatches, the resolver ignores the response and continues to wait. If on the other hand a reply arrives before the Hold-On time expires and validates based on the above tests, the resolver accepts the new reply and returns it to the client.

If the stub resolver receives no valid reply before the Hold-On timer expires, it returns the latest non-validating reply it observed. Doing so means that in the presence of significantly changed network conditions, users experience delay, but not inadvertent blocking of their access.

In most cases, the resolver will not wait until the Hold-On timer timing out; it will stop waiting upon receipt of a legitimate response. Thus, generally this approach will not cause extra delay, except in the case that network conditions have changed such that legitimate replies now return sooner and without DNSSEC protection.

IV. Feasibility Assessment

To assess the viability of our approach, we investigate the phenomenon of observing multiple replies for a single DNS query in both a censored network and a non-censored network. In the latter, we look at whether normal DNS traffic generates such replies; that is, whether Hold-On and validation could cause significant false positives. In the censored network, we assess how different the injected replies appear from the legitimate ones, which indicates whether the approach could suffer from significant false negatives.

A. Observation in an uncensored network

We can view use of the Hold-On approach as a form of anomaly detector, looking for a condition that represents an attack. Although it is clear that a packet-injection based DNS attack must create an anomaly where the client receives two distinct replies, we must ensure that normal DNS traffic does not generate these anomalies, as, in some cases, there may

be no effective resolution beyond simply noting the attack and returning no valid answer if it proves impossible to heuristically distinguish an attacker's packet from a legitimate non-DNSSEC signed reply. If the resolver simply ignores replies it cannot validate (and returns the last such, if no valid replies are received), then such anomalies arising in legitimate traffic will not in fact cause any problems. If, however, the resolver flags such replies as reflecting an attack, then these false positives will incur a degree of collateral damage.

We developed a Bro [23] IDS policy script to directly detect anomalous secondary DNS replies. This script operates by tracking all DNS requests and matching replies, checking any subsequent reply that arrives within a 1-minute timeout[2] to determine whether the number of records in the reply and the contents of each are unchanged. We validated that this script accurately detects attack packets using traces of injected packets we captured by sending DNS query requests that transited a network that uses DNS-based packet-injection censorship.

We ran this script against 6 days of normal DNS traffic captured at ICSI's border, consisting of 11,700,000 DNS requests.[3] During this period we observed no DNS anomalies that would create a false positive, only deliberate testing intended to trigger a DNS censorship system.

Running on a 1.5 hour trace gathered in August 2011 at the UC Berkeley campus border (a total of 15.2M DNS transactions,[4] both inbound and outbound), we observed two benign authorities that triggered the basic anomaly detector. The first server, an authority server for the BBC, returned two distinct replies for the same query for several names. Although distinct in value, both values were within the same /24 subnet.

The second, an authority for businessinsider.com, returned two values for the same query. The first reply was a CNAME to an external domain with the root authority information included in the reply, while the second was a SERVFAIL containing the same CNAME but no authority or additional fields, triggering the alert.

We also observed both multiple incidents of DNS censorship (caused by local users configured to use resolvers in a censored country) and a few false-positives due to script bugs that would not disrupt a Hold-On resolver.

B. Observation in a censored network

To assess potential false negatives, we trigger a DNS censorship system to inject DNS replies with sensitive domain names (such as twitter.com). We generated these measurements from within the censored network, communicating with destinations outside the censored network. To differentiate the legitimate from the injected replies, we first query a non-existent DNS server outside the censored network with sensitive names, and we receive only injected replies. We then query an open DNS server with non-sensitive names (such as www.mit.edu), by which we receive only legitimate replies.

[2]We chose a longer timeout to be conservative in this analysis, attempting to detect potential anomalies that would not affect a resolver using Hold-On.

[3]We excluded lookups issues by an ICSI measurement tool.

[4]Excluding a known high-volume DNS crawler used for research.

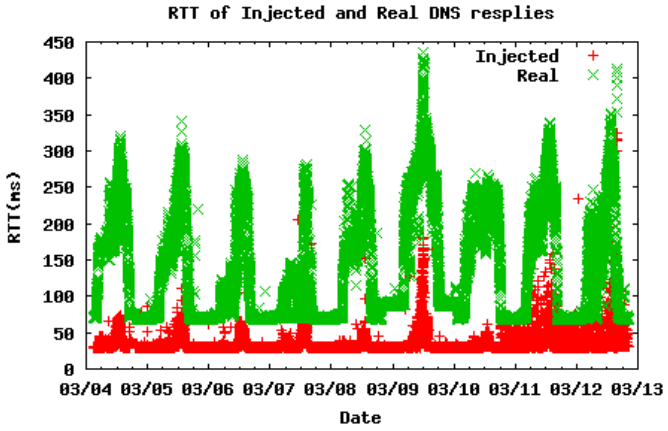

Fig. 2. Comparison of arrival times for legitimate packets (green cross) and packets injected by censor (red plus)

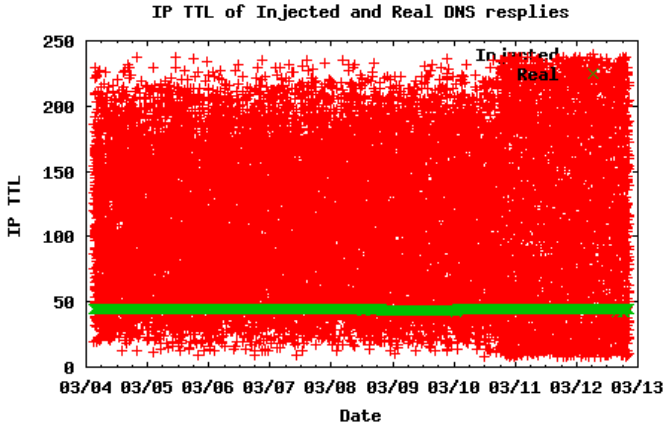

Fig. 3. Comparison of TTLs for legitimate packets (green cross) and packets injected by censor (red plus)

With this method, we collected a data trace including $\approx$ 100,000 queries and corresponding replies over 9 days. Figures 2 and 3 show comparisons of RTTs and TTLs observed of legitimate DNS packets and injected packets by the DNS censor. It appears not difficult to identify the legitimate packets from injected. Most injected packets arrive much earlier than legitimate ones because the injector and the client reside within the same ISP, while the DNS server resides in another country. We found the values of IP TTL from the legitimate DNS responses are quite stable over a period of 9 days (either 44 or 42), but the TTL value of the injected packets varied in the range of [0–255], presumably to avoid simple filtering.

In another 10-hour trace, we select one pair of (RTT, TTL) every 5 minutes, and use this as the expected RTT and TTL to validate other packets in the following time window. In our experiment, we change the threshold of TTL and RTT to evaluate the false positive rate and false negative rate, as shown in Table I. For example, if we set the threshold

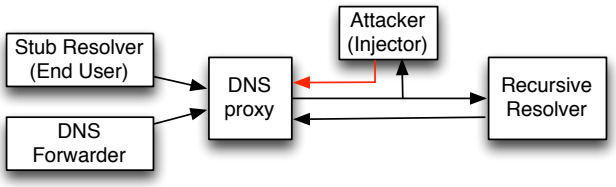

Fig. 4. Environment of DNS proxy

of TTL to 1 (that is, the reply is valid only if $TTL \in [expected\_TTL-1, expected\_TTL+1]$) and set the threshold of RTT to *0.5 · expected RTT* (that is, the reply is valid only if it does not arrive *0.5 · expected RTT* earlier than expected), then the approach does not generate any false positives or negatives.

TTL threshold	RTT threshold	FP (%)	FN (%)
0-2	0.5	0	0
3	0.5	0	0.01
4	0.5	0	0.06
5	0.5	0	0.07
6	0.5	0	0.10
7	0.5	0	0.11
2	0.1	5.96	0
2	0.2	1.53	0
2	0.3-0.8	0	0
2	0.9	0	0.31

TABLE I
FALSE POSITIVE (FP) AND FALSE NEGATIVE (FN) RATES CORRESPONDING TO DIFFERENT THRESHOLDS FOR IP TTL AND RTT DIFFERENCES.

V. IMPLEMENTATION AND EVALUATION

We implemented a DNS proxy to explore how Hold-On works in practice. The proxy operates as a DNS forwarder that aims to protect against DNS injection by on-path adversaries, as illustrated in Figure 4.

A. Design and implementation of a DNS proxy

To estimate the expected RTT and TTL to/from the remote recursive resolver, the proxy issues requests upon start-up for non-sensitive names.[5] To estimate the RTT, the resolver queries the same name multiple times, selecting the minimum of RTT observed. The resolver excludes the first query, because it might include additional time consumed by the server to resolve the name recursively, rather than answering from its cache. The expected TTL(s) should typically remain constant, but could vary due to routing changes.[6] We assume that the set of expected TTLs does not vary in a measurement period (see below). In our current implementation, the set has only one value. During its normal operation, a separate thread repeats this measurement (see § IV-B) periodically (such as every 5 minutes) and updates the expected RTT and TTL values adapted to potential change of network status.

[5]It could instead simply monitor initial queries for duplicate replies, and formulate its estimates from those that engender only a single reply. Doing so would also help with combating injection from attackers who have different goals than censorship.

[6]A potentially pathological case would be replies that vary across a set of arriving TTL values due to the use of per-flow load-balancing that causes different replies to take different routes.

Algorithm 1 Hold-On and Validation for DNS Proxy

```
Timeout ← 5
while GetDNSRequestFromClient(request) do
  retry ← 1; gotAnyReply ← false
  repeat
    ForwardRequestToResolver(Resolver, request);
    StartHoldOnTimer(retry · Timeout);
    while NOT Timeout and GetDNSReply(replyPkt)
    do
      gotAnyReply ← true { from server or injector}
      if ValidateDNSSEC_OK(replyPkt) then
        SendDNSReplyToClient(replyPkt.msg)
        StopHoldOnTimer()
        return
      else if ValidateTTL_OK(replyPkt.ipTTL)
      and ValidateRTT_OK(replyPkt.RTT) then
        SendDNSReplyToClient(replyPkt.msg)
        StopHoldOnTimer()
        return
      else
        DropAndLog(replyPkt)
      end if
    end while
    retry ← retry + 1
  until retry == 3
  if gotAnyReply then
    {No valid reply, return the latest non-validating reply}
    SendDNSReplyToClient(replyPkt.msg)
  end if
end while
```

Algorithm 1 details how the proxy processes with DNS requests and replies. When the proxy receives a DNS request from its client (end user or DNS forwarder), it forwards the request to the remote recursive resolver and starts the Hold-On timer. We set the initial value of the timer to 5 seconds; if no legitimate reply after the timer expires, we reset the timer to 10s for the second try, and similarly to 15s for the third try.

If the proxy receives a DNS reply (from either the remote recursive resolver, or an injector), it validates both TTL and RTT against the expected values (the expected TTLs could include multiple values because of multiple paths to the resolver). If the request is DNSSEC enabled, the corresponding reply should also be checked with DNSSEC options (not implemented yet in our prototype). For DNSSEC-disabled requests, ValidateDNSSEC_OK always returns **false**. ValidateRTT_OK and ValidateTTL_OK return true if:
$expected\_RTT - replyPkt.RTT < 0.5 \cdot expected\_RTT$
$replyPkt.ipTTL \in expectedTTLs$

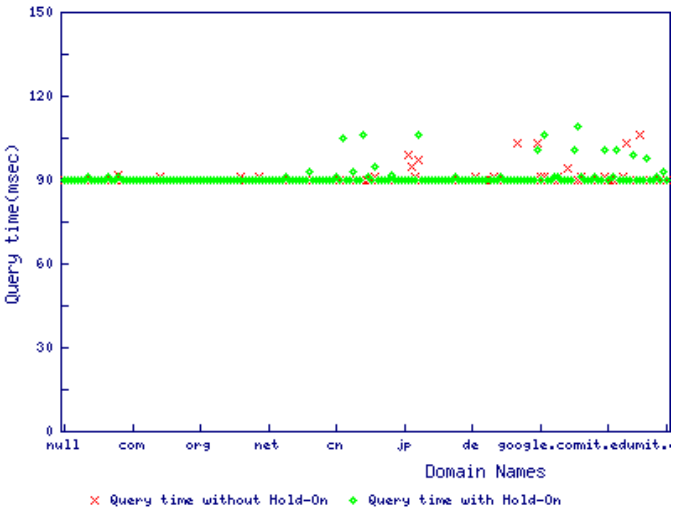

Fig. 5. For cached names, compare of the query time with Hold-On enabled (green square) and Hold-On disabled (red cross). Most of the points in the two set overlap, so only a green line is shown.

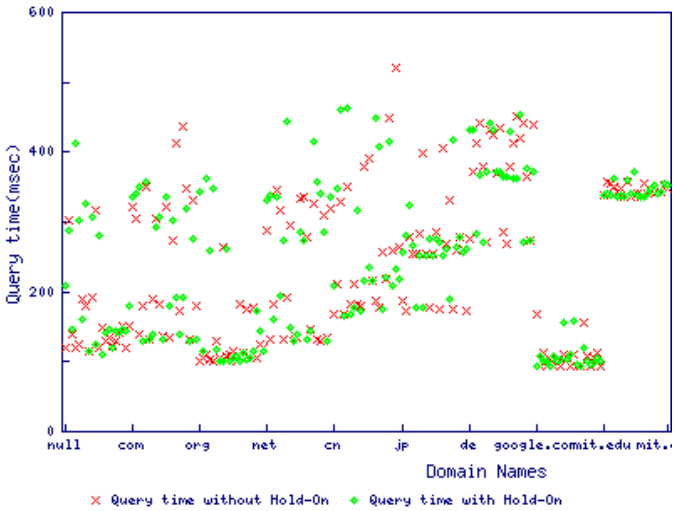

Fig. 6. For uncached names, compare of query time with Hold-On enabled (green square) and Hold-On disabled (red cross).

B. Evaluation

To evaluate the delay increased by Hold-On, we use two sets of domain names for cached or uncached queries. For the first set, we used a fixed prefix ("www") with different levels of domains appended (".” or "null", "com", "org", "net", "cn", "jp", "de", "google.com", "mit.edu"). For the first set, we query each name first, let the recursive resolver cache the result, and then measure the query time for subsequent queries. As shown in Figure 5, the time needed for each query approximates the RTT.

We generated the second randomly using a nonsense prefix again with different level of domains appended. We query each name only once.

Because the resolver must fully resolve the name for each query, the query time includes RTT (from the proxy to the recursive resolver) as well as the subsequent time consumed by the resolver (from recursive resolver to authoritative name servers). As shown in Figure 6, the time varies considerably for different level of domain. For example, it is much faster to resolve "nonsense12847.google.com" than "nonsense2132323.de" because the resolver (in our case, Google Public DNS) has a low-latency path to a google.com authority, but a higher latency path to the .de authority.

As shown in Figure 5 and 6, in both cases (cached or uncached query) Hold-On and validation do not introduce any apparent delay.

VI. Discussion

The current Hold-On implementation operates as a stub resolver to a known-uncensored remote recursive resolver, which enables accurate initial measurement of RTT and TTL to enable subsequent detection of unexpected replies. Without DNSSEC, it relies on attack packets exhibiting significant differences in IP TTL or RTT in order to distinguish them from legitimate replies.

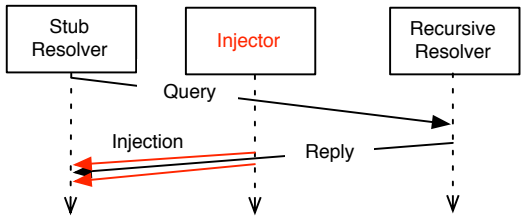

Fig. 7. The attacker carefully crafts the packets' TTL and injects them with the expected timing

If the attacker carefully crafts the attack packets' TTLs, and only injects them just before the likely arrival of the legitimate reply (see Figure 7), our Hold-On technique without DNSSEC will fail to distinguish between the legitimate reply and the attack packet.[7] However, the resolver can still detect that an attack has likely occurred (due to the differing responses). If the resolver cannot correctly determine the legitimate response, it should conservatively not return any answer—though doing so still enables attackers to impose denial-of-service. Hold-On can provide a robust defense against on-path injection only when combined with DNSSEC.

Extending our implementation to recursive resolvers will take some additions. However, most recursive resolvers already maintain estimated RTTs to different authority servers in order to select between nameservers. This timing data, combined with tracking of TTLs, can enable detection (if not protection) of injected packets.

In addition, as developed so far, our particular Hold-On mechanism does not suffice to protect against on-path adversaries operating in environments that users only occasionally

[7]The adversary must take the risk that the injected packets arrive later than the legitimate ones.

visit, such as public WiFi hotspots; we lack knowledge of "non-sensitive" domains to look up in order to obtain estimates of RTT and TTL in the presence of non-injection. In such environments, we can in principle instead look for symptoms of injection (not only for DNS, but potentially for DHCP, ARP, and TCP), alert the user in some fashion, and terminate any associated connections. Clearly, developing a robust, workable approach along these lines will take significant investigation.

Finally, a recursive resolver should also continue to listen for replies even after it processes and forwards an answer, in order to detect attacks. If the resolver detects an attack, it could revert to a more paranoid mode (performing active measurements of RTTs and TTLs for all new resolvers to detect subsequent attacks). In addition, it can flush all cache entries associated with the attack if it is unable to distinguish between the legitimate reply and the attack. (If it can tell that the later reply is the legitimate one, the resolver can use it to replace the cache entry.) Doing so limits the damage from the injected packet to the improperly returned question. It also means that the user, upon noticing a failure or redirection for a potentially censored site, can simply attempt to reload the site, this time receiving the valid reply for their answer, circumventing the censorship. Finally, the extended recursive resolver can also itself issue an additional response (with the second reply's contents) in response to the client's query, so that any client or stub that also detects mismatched replies will be aware of the attack.

VII. Conclusion

Packet injection by on-path adversaries poses a particular threat against DNS. One particular use of such injection comes from those building DNS-based censorship tools. Even DNSSEC-validating implementations may be vulnerable, as injected packets can cause denial-of-service if the resolver fails to wait for the legitimate reply before returning "Bogus".

Such adversaries cannot however block the receipt of the legitimate DNS reply in addition to their injection. We show that this artifact—differing replies to the same question—appears to occur only very rarely outside of an actual attack. Thus, stub resolvers can use its presence to detect when packet injection occurs.

In addition, unless censors take additional steps, imperfections in their packet injection tools can allow resolvers to not only detect injection attacks, but also to potentially distinguish between legitimate replies and injected answers based on artifacts in the IP header's TTL field and in the round-trip time when receiving replies. Accordingly, we propose that resolvers should utilize a *Hold-On* period, waiting for additional replies. If a reply arrives too early, with an unexpected IP TTL, or fails to validate (if DNSSEC validation occurs), the stub waits for the potential arrival of a subsequent legitimate reply before proceeding. For censors who take steps to match the TTL and RTT of their injections with those expected for legitimate replies, Hold-On still allows detection that injection has occurred; and, for DNSSEC-enabled resolution, prevents the censor from imposing denial-of-service on obtaining the correct reply.

Finally, we developed a DNS forwarder that implements Hold-On, and demonstrated that this forwarder is effective at distinguishing legitimate replies from those injected by a widely deployed network censorship system. Our evaluation found that use of such forwarding imposes minimal additional latency on undisrupted DNS queries, and thus Hold-On has promise for enabling more robust DNS resolution in the face of on-path censorship.

VIII. Acknowledgments

This work was sponsored in part by NSF grants 0831780, 0905631, and CNS-1015835. Special thanks to Robin Sommer for assistance with the UC Berkeley network traces.

References

[1] G. Lowe, P. Winters, and M. L. Marcus, "The Great DNS Wall of China," Dec. 2007. http://cs.nyu.edu/%7Epcw216/work/nds/final.pdf.
[2] P. Mockapetris, "Domain Names—Concepts and Facilities, RFC 1034." http://www.ietf.org/rfc/rfc1034.txt.
[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol Modifications for the DNS Security Extensions," 2005. http://www.ietf.org/rfc/rfc4035.txt.
[4] S. M. Bellovin, "Using the Domain Name System for System Break-ins," in *Proceedings 5th USENIX Security Symposium*, 1995.
[5] D. Kaminsky, "Black ops 2008: It's the end of the cache as we know it," *Black Hat USA*, 2008.
[6] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee, "Increased DNS Forgery Resistance Through 0x20-bit Encoding: security via leet queries," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 211–222, ACM, 2008.
[7] N. Weaver, R. Sommer, and V. Paxson, "Detecting Forged TCP Reset Packets," in *NDSS'09*, 2009.
[8] P. Mockapetris, "Domain Names—Implementation and Specification, RFC 1035." http://www.ietf.org/rfc/rfc1035.txt.
[9] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirement," 2005. http://www.ietf.org/rfc/rfc4033.txt.
[10] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource Records for the DNS Security Extensions," 2005. http://www.ietf.org/rfc/rfc4034.txt.
[11] D. Dagon, N. Provos, C. Lee, and W. Lee, "Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority," in *Proceedings of Network and Distributed Security Symposium*, 2008.
[12] P. Roberts, "Chinese DNS Tampering A Big Threat To Internet Security," 2010. https://threatpost.com/en%5Fus/blogs/chinese-dns-tampering-big-threat-internet-security-112410.
[13] J. G. Høy, "Anti DNS Spoofing-Extended Query ID (XQID)." http://www.jhsoft.com/dns-xqid.htm, 2008.
[14] D. Atkins and R. Austein, "Threat Analysis of the Domain Name System (DNS)," *RFC3833*, 2004.
[15] B. Hubert and R. Mook, "Measures for Making DNS More Resilient against Forged Answers, RFC 5452," 2009. http://tools.ietf.org/html/rfc5452.
[16] R. Clayton, S. Murdoch, and R. Watson, "Ignoring the great firewall of china," in *Privacy Enhancing Technologies*, pp. 20–35, Springer, 2006.
[17] J. R. Crandall and E. Barr, "Conceptdoppler: A weather tracker for internet censorship," in *14th ACM Conference on Computer and Communications Security*, 2007.
[18] "Google Public DNS." http://code.google.com/speed/public-dns/.
[19] "OpenDNS Homepage." http://www.opendns.com/.
[20] "DNS Survey: Open Resolvers." http://dns.measurement-factory.com/surveys/openresolvers.html.
[21] P. Albitz and C. Liu, *DNS and BIND, 5th Edition*. O'Reilly, 2006.
[22] "DNS: The forwarding timeout value should be 2 to 10 seconds," 2010. http://technet.microsoft.com/en-us/library/ff807396(WS.10).aspx.
[23] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.