A Peek Into the DNS Cookie Jar

An Analysis of DNS Cookie Use

Jacob Davis[1,2] [0000-0002-2372-9203] and Casey Deccio[2] [0000-0003-0938-375X]

[1] Sandia National Laboratories, Livermore CA 94551, USA
jacdavi@sandia.gov
[2] Brigham Young University, Provo UT 84602, USA
{jacobdavis,casey}@byu.edu

Abstract. The Domain Name System (DNS) has been frequently abused for Distributed Denial of Service (DDoS) attacks and cache poisoning because it relies on the User Datagram Protocol (UDP). Since UDP is connection-less, it is trivial for an attacker to spoof the source of a DNS query or response. DNS Cookies, a protocol standardized in 2016, add pseudo-random values to DNS packets to provide identity management and prevent spoofing attacks. In this paper, we present the first study measuring the deployment of DNS Cookies in nearly all aspects of the DNS architecture. We also provide an analysis of the current benefits of DNS Cookies and the next steps for stricter deployment. Our findings show that cookie use is limited to less than 30% of servers and 10% of recursive clients. We also find several configuration issues that could lead to substantial problems if cookies were strictly required. Overall, DNS Cookies provide limited benefit in a majority of situations, and, given current deployment, do not prevent DDoS or cache poisoning attacks.

Keywords: Internet Measurement · DNS · DNS Cookies · DNS Security

1 Introduction

The Domain Name System (DNS) is an essential backbone of the internet used to translate domain names to Internet Protocol (IP) addresses. Since its inception in the 1980s, the DNS has been reliant on the User Datagram Protocol (UDP). While UDP has a major benefit of speed, its lack of identity management is easily exploitable. Off-path attackers can spoof UDP packets to pretend they, or a victim, are the source of the packet.

There are two major attacks utilizing spoofing. The first is cache poisoning, wherein an attacker sends malicious responses pretending to be a legitimate server. If successful, the victim is unknowingly directed towards a malicious IP address. The other attack is a DNS reflection attack. This attack is carried out by sending many DNS queries with the victim's IP address as the spoofed source and results in the victim being flooded with unsolicited response traffic—a form of distributed denial-of-service (DDoS).

Both cache poisoning and reflection-based DDoS attacks exploit the lack of verification inherent with UDP. In an attempt to solve this issue, and provide

identity management in the DNS, a new protocol, known as DNS Cookies, was standardized through the Request for Comments (RFC) process in 2016 [11]. With DNS Cookies, both client and server include a cryptographic identifier (the cookie) in their DNS messages which can then be verified in future messages. An off-path attacker is unable to learn the cookie values and thus cannot feasibly spoof them.

Since 2016, DNS Cookies have become increasingly common and are supported by many open-source DNS software vendors. However, to the best of our knowledge, no research has been done to quantify the level of support for cookies. The major contribution of this paper is a **study of client- and server-side support for DNS cookies**—the first such measurement of its kind. Additionally, we analyze DNS Cookie enforcement to see if any client or server rejects illegitimate DNS messages based on cookies. While clients and servers may be exchanging cookies, there is no benefit unless a missing or incorrect cookie affects the server's response.

In this paper, we make the following contributions:

- We measure support for DNS Cookies in high-demand authoritative DNS servers and open resolvers Internet-wide; we find that 30% of servers fully support cookies, and only 10% of recursive clients send cookies.
- We analyze the DNS Cookies we observed and discover several potential misconfigurations, such as inaccurate server clocks, some of which could break implementations.
- We examine the behavior of DNS clients and servers when encountering missing or illegitimate cookies and find that 80% of clients do not reject responses when they should and that 99% of servers handle these situations in the least restrictive manner by responding indifferently.
- We discuss the path forward for wider DNS Cookie adoption and possible solutions for enforcing the use of cookies.

Overall, our work, which is the first to measure DNS Cookies in the wild, reveals a low level of adoption and minimal enforcement of DNS Cookies. We believe that DNS Cookies have the potential to benefit the DNS, but greater adoption and strategies for enforcement are required.

Artifacts: The source code and datasets used to produce this paper can be found at the following link: `https://imaal.byu.edu/papers/2021_pam_dns_cookies/`.

2 Background

The Domain Name System (DNS) is primarily used to convert domain names (e.g. `example.com`) to Internet Protocol (IP) addresses (e.g. `192.0.2.1`) [18,19]. There are three components in the DNS: stub resolvers, recursive resolvers, and authoritative servers.

Stub resolvers are typically associated with end-devices such as a phone or desktop. To visit a given domain, a stub sends a DNS query to its configured recursive resolver. The recursive resolver can respond to the query immediately if the answer is cached. Otherwise, it queries several authoritative servers systematically until it obtains the answer.

The DNS continues to utilize the User Datagram Protocol (UDP) as its primary transport protocol. UDP does not provide identity management and therefore does not protect against spoofing attacks, wherein an attacker impersonates a client or server by using their IP address as the source.

One attack that utilizes spoofing to impersonate an authoritative server is DNS cache poisoning. With cache poisoning, an attacker can respond to a client with a malicious IP address, causing that client, and all who rely on its cache, to be redirected to the malicious IP.

Due to the severity of a successful cache poisoning attack, several measures have been encouraged to reduce the chance of a successful cache poisoning. These include source port randomization [15] and `0x20` encoding (randomized capitalization) [6]—both of which require only changes to client-side software. Another avenue would be for a client to use DNS-over-TCP [10], DNS-over-TLS (DoT) [14], or DNS-over-HTTPS (DoH) [13]. These three protocols all provide the identity management inherent in the TCP handshake, and DoT and DoH are showing increased adoption [9,17]. However, they result in increased latency [7]. A final approach, which avoids identity management altogether, is cryptographically authenticating DNS responses. This strategy is employed by DNSCurve [1] and the DNS Security Extensions (DNSSEC) [3,4,5]. Neither of these methods has seen widespread adoption.

Another attack that exploits the lack of identity management in UDP and the DNS is distributed denial-of-service (DDoS) attacks. Here the attacker impersonates the victim's client and sends many DNS queries. This results in traffic being reflected off of DNS servers and the victim being flooded with unsolicited response traffic. Past attacks have reached traffic volumes of 300Gbps to 1.2Tbps and are capable of affecting major services such as Amazon and Netflix [20,12]. Both of these attacks can have major effects but can be prevented with some form of identity management.

DNS Cookies [11] are designed as a lightweight mechanism that provides identity management at a strength similar to TCP, but without the latency burden. They are included in DNS messages as a `COOKIE` option inside the Extended DNS (EDNS) `OPT` resource record [8]. Both the client and server in a given communication can provide a plain-text cookie in their DNS messages. The client can then verify that the server includes the client cookie (i.e., provided by the client) in future communications—and vice-versa—to ensure that messages have not been spoofed by an off-path attacker. An example of this process is shown in Figure 1. DNS Cookies do not provide protection against on-path attackers, but should still provide substantial benefit to securing the internet as a whole.

Client cookies are 8 bytes in length and are used to prevent cache poisoning by enabling the client to verify the server's identity. A stub or recursive resolver can

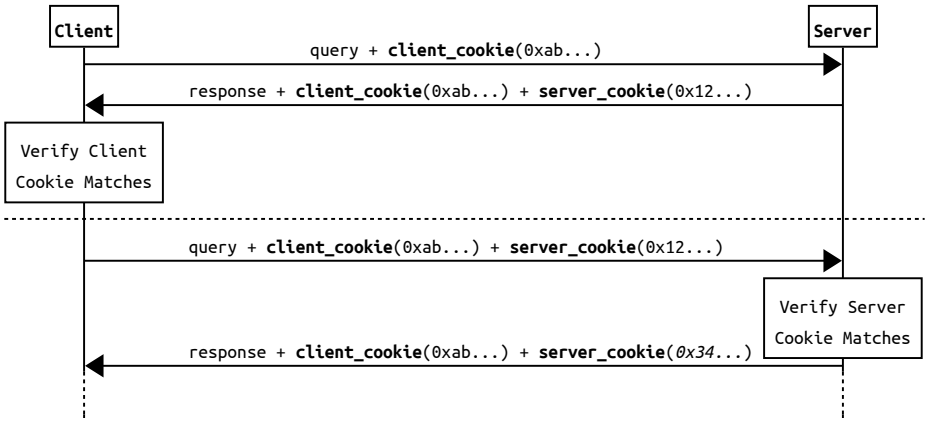

Fig. 1. An example communication using DNS Cookies. Here the client starts from a fresh state and reuses its cookie whereas the server generates a new cookie per query.

include a DNS client cookie in their queries and only accept a response containing the cookie. The suggested implementation for generating a client cookie is to use a cryptographic hash of the *(Client IP |Server IP |Client Secret)*. More recent suggestions remove the inclusion of the Client IP as it may not be known at the time of generation [21]. Regardless, a client should use a unique cookie per server and should not reuse a cookie across IP addresses as this would enable tracking the client.

A server cookie ranges in size from 8 to 32 bytes and is used to confirm a client's identity, in turn preventing reflection-based attacks. Authoritative and recursive servers may choose to send a server cookie when responding to a query with a client cookie in it. Clients should then include this cookie in future queries to verify their identity. If a server receives a query without a valid server cookie they may enforce cookie use by responding with the `BADCOOKIE` response code (rcode), a valid server cookie, and no DNS answers. The specification suggests that a server cookie consists of a 4-byte nonce, 4-byte timestamp, and an 8-byte cryptographic hash of the *(Server Secret |Client Cookie |Nonce |Time |Client IP)*. The time field results in a new cookie for every request and makes rejection of outdated cookies easy. Additionally, the server does not need to save any state to verify a cookie as the nonce and timestamp are provided in plain-text.

In 2019 an Internet draft was created to standardize the format for DNS Cookies to allow interoperability between different DNS software [21]. Of note, server cookies were visibly changed as the nonce was replaced with a version and reserved field.

3 Support for DNS Cookies

Here we establish a baseline measurement for DNS Cookie usage from the perspective of both clients and servers. We analyze DNS server-side cookie behavior,

which includes both authoritative DNS servers and recursive resolvers in their "server" role to clients. For this analysis, we classify varying levels of support: EDNS capability (via the inclusion of an option (`OPT`) record in a response), echoing of a sent client cookie (only), and full support with a returned server cookie. While echoing a client cookie is not a specified option in the protocol, it does still protect the client. We also measure cookie usage of recursive resolvers in the "client" role in connection with queries to authoritative servers under our control. An analysis that included all perspectives would have included DNS Cookie use by stub resolvers in their communications with DNS recursive resolvers. However, that data is available only to recursive server operators, so we were unable to perform an analysis of stub resolver behavior with respect to DNS Cookies.

3.1 Server-Side Cookie Support

We queried a set of open recursive resolvers and two sets of authoritative servers to measure DNS Cookie support for "servers".

To generate a set of recursive resolvers to test, we issued a DNS query (for a domain we control) to every IPv4 address. We classified an IP address as a recursive resolver if it 1) queried our authoritative server or 2) responded to our query with the recursion available (`RA`) flag set and a response code of either `NOERROR` or `NXDOMAIN`. This data was collected from September 24–26, 2020. In total, we identified 1,908,397 open recursive resolvers.

For authoritative servers, we analyzed servers authoritative for the top 1 million Alexa domains [2] (actually 770,631 domains) and servers authoritative for the 1,509 top-level domains (TLDs) [16] (including the root servers). All data was collected on September 30, 2020, using the latest Alexa file and root zone available. The names and IP addresses (IPv4 and IPv6) for each domain in the collective lists were determined through 1) a lookup of type `NS` (name server) for the domain and 2) a lookup of type `A` and `AAAA` (IPv4 and IPv6 address, respectively) for each name returned in the `NS` query response. In total, we recorded 157,679 IP addresses for the Alexa sites and 6,615 for the TLDs.

To identify support for cookies, we issued up to 6 DNS queries to each server—stopping early if we received a response with a server cookie. We included the same client cookie in every query. During these queries, we experienced errors with 48% of resolvers, likely due to high churn. In particular, queries for 32% of resolvers timed out, and for 16% of resolvers, we received a response from a different IP address (often Cloudflare's `1.1.1.1`) than we had queried. Removing these cases leaves us with 999,228 error-free resolvers. For authoritative servers, queries to 6,724 (4.3%) of Alexa IPs resulted in an error, as did queries to 58 (0.88%) TLD IPs. The errors associated with querying authoritative servers primarily consisted of time outs (98% of Alexa errors and 100% of TLD errors), though there were a handful of malformed packets or unexpected responses. We report all of our results as percentages of communications with error-free servers.

EDNS, which is a prerequisite for cookies, was supported (as evidenced by an `OPT` record in responses) by 699,402 (70%) of recursive resolvers, 147,878 (98%) of Alexa IPs, and 6,557 (100%) of TLD IPs. The client cookie that we sent in our queries was returned by 208,526 (21%) of recursive resolvers, 48,262 (32%) of Alexa IPs, and only 1,249 (19%) of TLD IPs. The remaining servers returned a response that either did not include a `COOKIE` EDNS option or included a client cookie that did not match the one we sent. Servers that included a server cookie in their response (this implies the inclusion of a client cookie, by specification) include: 167,402 (17%) of open resolvers, 43,649 (29%) of Alexa IPs, and all 1,249 of the TLD IPs that returned the correct client cookie. However, 14 resolver IPs and 5 Alexa IPs returned a `COOKIE` option with a server cookie of all zeroes. The Alexa and TLD IPs that returned server cookies were collectively authoritative for 26,629 domains and 373 zones respectively.

Of note, 93 Alexa IPs and 41 resolvers IPs responded with a client cookie that did not match the one we sent. For 5 Alexa IPs and 22 resolvers IPs, the value of the client cookie returned was off by only one byte—the fourth most significant byte. An additional 5 Alexa and 14 resolver IPs replied with zeroed out client cookies. A single TLD IP, one of three servers authoritative for the `gm` TLD, returned a `COOKIE` option with all zeroes for both the client and server cookies. The remaining unexpected responses did not follow a discernible pattern.

Overall we observe high EDNS support (70% of resolvers and >98% of authoritative servers). However, cookie support is much lower. While nearly one-third of Alexa IPs fully supported cookies, less than 20% of TLD IPs and recursive resolvers did. As a result, there are still more than 100,000 authoritative servers and 800,000 recursive resolvers that can be used for reflection attacks because they lack a mechanism for validating client identity.

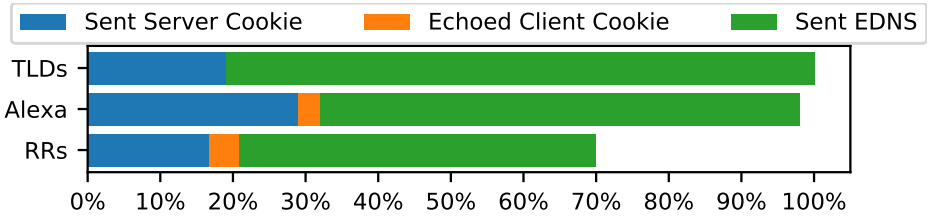

Fig. 2. Incremental support for DNS Cookies across the three datasets of recursive resolvers, TLD authoritative servers, and the top 1m Alexa authoritative servers. Servers in the leftmost group fully support DNS Cookies.

3.2 Client-Side Cookie Support in Recursive Resolvers

During our measurements of resolvers supporting the "server" role of cookies, we also measured their support for DNS Cookies while acting as a "client". Each query we issued consisted of a special domain name hosted at authoritative

servers under our control. We recorded each incoming query for the domain name we were using and responded with `NXDOMAIN` and full DNS Cookie support. We observed queries to our authoritative servers from 93,395 unique IP addresses, of which 8,471 (9.1%) sent at least one response that contained a `COOKIE` option.

During this measurement, we encoded the IP address of the recursive resolver that we queried in the domain name. This reveals that 1,552,397 unique resolvers queried our backend through the 90,000 IP addresses. This discrepancy may be due to forwarding, as 56% of resolvers were represented by only 1000 IP addresses. In particular, Google and Cloudflare handled queries for 36% and 7.0% of resolvers respectively.

In our measurement for recursive resolvers sending cookies, we found client cookie support to be minimal. Of resolvers that queried our authoritative servers directly, only 9.1% of over 90,000 IPs sent a cookie. This is potentially alarming as these resolvers are not using cookies for cache poisoning protection. While they may employ other methods, DNS Cookies offer an extra layer of defense.

4 Server Cookie Analysis

For our measurement of DNS server cookies, we expand the datasets from 3.1. For each IP address we found to be sending server cookies, we sent an additional 60 queries. These queries were broken into 3 subsets: the first 20 queries never included a server cookie, the next 20 included the first server cookies received, and the final 20 included the latest server cookie we had received. Each subset had a 1-minute pause after the first 10 queries, before issuing the final 10 queries.

Valid server cookies may be anywhere from 8 to 32 bytes in length. Of all of the cookies we received, >99% were 16 bytes in length.

4.1 Dynamic Cookies

Many server cookies are dynamic: changing consistently due to the inclusion of a timestamp (representing seconds since UTC). This follows the suggested implementation in the RFC. Additionally, the newer format of interoperable cookies includes a timestamp in the same position.

As a result, we classify a cookie as dynamic if bytes 5–8 represent a time within a window of 1 hour in the past and 30 minutes in the future compared to the current time of our querying machine (NTP synchronized). [3]

Over 99% of authoritative servers and 83% of recursive resolvers that sent server cookies used at least one dynamic cookie.

Timestamps We first consider the timestamps being used in dynamic cookies. We are primarily interested in three unusual patterns: timestamps consistently

[3] The chance of a non-dynamic cookie being classified as dynamic is extremely small. Our window size accepts only 5,400 values out of the 4.3 billion possible values in the 32 bit field.

off by more than a minute, cookies that are "sticky" for short periods, and slow-moving timestamps that update on a fixed interval of 10 or more seconds.

For each dynamic server cookie, we compared the timestamp with the current time of the querying machine (i.e., the client), which was NTP-synchronized: $ts_{diff} = ts_{cookie} - ts_{client}$. We consider a server cookie's timestamp to be *accurate* if $|ts_{diff}| \leq 5s$. This generous window accounts for any network delays. We consider a timestamp to be *significantly out-of-sync* if $|ts_{diff}| > 60s$. Finally, we classify "sticky" and slow-moving clock servers based upon the number of distinct values of ts_{diff} since this tells us that the ts_{cookie} remained static while ts_{client} advanced. "Sticky" servers are defined by having 8 or more distinct values in one of the 3 subsets of queries and less than 3 distinct values in another. Slow-moving clocks are defined by not being sticky and having 10 or more distinct values across all cookies.

Table 1 summarizes the major findings for each IP address. Over 95% of IPs consistently returned server cookies with accurate timestamps. For 2.8% of IPs, the timestamps were significantly out-of-sync, likely due to an incorrect clock. While an incorrect clock is unexpected, it is inconsequential for cookies since the cookie value only matters to the server itself.

A category that is perhaps more interesting is IPs for which we observed a mix of cookie timestamp behaviors—some accurate and some significantly out-of-sync. For example, one IP returned cookies resulting in the following values of ts_{diff}: (*1 1 1 1 75 1 2 1 75 1...*). The timestamps for approximately one-fifth of the responses were consistently and significantly out-of-sync, while the remainder were accurate. This behavior is representative of a DNS server with five backend servers, one of which has a clock that is 75 seconds out of sync.

We additionally observe that some IPs use "sticky" cookies: cookies that remain static for short periods (typically 10 seconds) depending on the context. We observed two implementations of this. In the first implementation, cookies were sticky when our client was not querying with a server cookie. Once our client began sending server cookies, the server replied consistently with accurate timestamp cookies. We observed that 77 Alexa IPs and 775 resolver IPs followed this pattern. The second implementation acted in the opposite manner: the server replied with accurate timestamp cookies until our client sent one of the server cookies in a query. The server then made that valid cookie sticky and did not change it for a short period. We saw this pattern in only 12 Alexa IPs and 12 recursive IPs.

Our final category consists of slow-moving clocks: cookie timestamps that update on a fixed interval of 10 or more seconds. We classified 20 Alexa IPs and 4,413 recursive resolver IPs in this category. We observed that 3,296 recursive IPs had at least one timestamp off by more than 2 minutes and that 2,206 IPs displayed strictly increasing ts_{diff} values across every set of 10 queries. From this, we can gather that most recursive resolver IPs are using a slow-moving clock (possibly intentionally) with an update period of over 2 minutes.

The timestamps in DNS Cookies proved to be more interesting than originally expected. We found that most servers always return a cookie with a current

timestamp; however, some implementations purposely hold onto a cookie for a short period. We also discovered potential configuration issues with some backends of an IP having an incorrect clock. If cookies were to be enforced, clients may be intermittently rejected if they present that backend's server cookie to another backend, and the cookie was too far out-of-sync to be considered valid.

Table 1. Summary of timestamps found in server cookies returned by IPs. ts_{diff} represents the difference between the timestamp in the cookie and the querying computer's current time.

	Alexa	TLDs	RRs
All Cookies Accurate ($\|ts_{diff}\| \leq 5s$)	41,639 (96%)	1,225 (98%)	131,520 (95%)
All Cookies Out-of-Sync ($\|ts_{diff}\| > 60s$)	1,615 (3.7%)	17 (1.4%)	3,544 (2.6%)
Mixed Accurate & Out-of-Sync	66 (0.15%)	0 (0.0%)	2,980 (2.2%)
"Sticky" Cookies	89 (0.21%)	0 (0.0%)	787 (0.67%)
Slow-Moving Clocks	20 (0.05%)	0 (0.0%)	4,413 (3.2%)
IPs Using Dynamic Cookies	**43,345**	**1,246**	**138,865**

Interoperable Cookies Interoperable Cookies are designed to standardize the generation of cookies across varying backend implementations. We classified a server cookie as interoperable if the cookie started with `0x01000000` as specified in the RFC draft (a one-byte version field and three bytes reserved) and the timestamp field met the criteria previously mentioned.

Of the 43,737 Alexa IPs that returned a server cookie, 1,778 (4.1%) used interoperable cookies consistently. For TLDs, 92 (7.4%) of 1,249 IPs used interoperable cookies. No IP in either dataset sent a mix of standard and interoperable cookies across all of our queries.

For the 167,402 recursive resolver IPs that sent a server cookie, we found that 30,078 (18%) sent at least one interoperable cookie. However, we also found that 10,948 (6.5%) of IPs sent a mix of interoperable and standard dynamic cookies [4]. This behavior was unexpected as the primary purpose of interoperable cookies is to standardize cookies across all backend servers behind a single IP address.

Overall adoption of interoperable cookies was low in authoritative servers (under 10%), but partial support in recursive resolvers was higher at 18%.

4.2 Static Cookies

While the majority of cookies can be classified as "dynamic", a number of servers reused the same cookie. We classified a server as using static cookies if only a single cookie was used across our tests and the cookie did not contain a dynamic

[4] It is possible that we misclassified a standard cookie with a nonce of `0x01000000` as being interoperable. 9,990 of these IPs sent at least two cookies that appeared interoperable in response to our 60 queries.

timestamp. We identified 38 recursive resolvers that used a unique 32-byte cookie for the entire duration of our test. Similarly, 33 Alexa servers always replied with a single, unique 8-byte cookie.

We further analyzed IPs for 4 Alexa domains that sent static cookies: `ibb.co`, `pantip.com`, `postimg.cc`, and `wikipedia.org`. For each IP address authoritative for these domains we sent queries every minute for four days and additional queries with varying client cookies and client IP addresses.

Our results show that all four domains used the client IP address and client cookie in the creation of their server cookie because changing either of these variables affected the cookie they returned. Each also changed their cookie at the start of every hour, implying that they either changed their secret or that an hourly timestamp was considered in the calculation. Of note, the authoritative servers for two domains—`wikipedia.org` and `pantip.com`—returned the same server cookie, regardless of which server was queried for the domain. However, the servers authoritative for `ibb.co` and `postimg.cc` acted independently, implying separate server secrets or some other unique value per server.

5 The State of Cookie Enforcement

In this section, we explore how clients and servers handle unexpected behavior. We begin by demonstrating to clients and servers that our infrastructure supports cookies. We then perform tests with missing cookies, missing EDNS, or fake cookies. With this, we can see whether clients and servers will enforce cookies if they know the other party supports them. If not, cookies provide little value as an attacker could simply exclude cookies in their spoofed packets.

5.1 Client Handling of Unexpected Server Behavior

For this experiment, we forced the 1.5 million resolvers (with or without cookie support) found in section 3.2 to query our authoritative servers 6 times. We configured our authoritative server to respond differently depending on the query name it received. The response conditions we created are as follows (in order):

1. NORMAL: Respond with full cookie support: Correct client cookie and a server cookie—if the query included a client cookie.
2. NO-COOKIE: Respond with no `COOKIE` option.
3. BAD-ANSWER: Respond with the correct client cookie (if any), `BADCOOKIE` rcode, and an answer section.
4. BAD: Respond with the correct client cookie (if any), `BADCOOKIE` rcode, and *no* answer section.
5. NO-EDNS: Respond with no `OPT` record (i.e., no EDNS support).
6. FAKE: Respond with incorrect client cookie.

For each query, we made up to 3 attempts, as the stub resolver, to receive an answer. This experiment was run approximately one week after we discovered the 1.5 million IPs. As a result, we experienced a high churn and only saw

528,832 (34%) of IPs respond with both an answer and an rcode of NOERROR in our NORMAL condition. [5]

Responses with Missing/Invalid Client Cookies Of those resolvers from which we still received responses, 28,605 (5.4%) included a cookie in the NORMAL condition (or the intermediate IP did). For these IPs in the NO-COOKIE scenario, we surprisingly got a normal response from 23,979 (84%) IPs. Of those with bad responses, 3,625 (13%) had a SERVFAIL rcode and an additional 909 (3.2%) timed out. For the NO-EDNS queries, we saw similar numbers compared to those who sent cookies: 24,798 (87%) responded to our stub resolver normally, 2,495 (8.7%) responded with SERVFAIL, and 1,236 (4.3%) timed out.

Finally, in the FAKE category, we began to see more rejection. This test was performed a day after NO-EDNS and as a result, there was more churn and some servers may have stopped sending EDNS since we appeared to not support it. We recorded 27,079 IPs which sent a cookie in a NORMAL query directly preceding this test. We saw a much lower percentage of acceptance here with only 5,115 (20%) responding to the stub resolver normally. Most failure is split between SERVFAIL with 10,059 (40%) of IPs and time outs with 9,564 (38%) of IPs.

The specification for DNS Cookies states that a client must discard a response with an invalid client cookie or a missing cookie when one is expected. However, we observed that 20% of recursive clients did not reject invalid cookies and that over 80% of clients did not discard responses that were missing a cookie when one should have been present (as demonstrated to the client in a previous query). This means that a majority of recursive clients may still be susceptible to cache poisoning attacks because a response without EDNS or a DNS COOKIE option is accepted as easily as a legitimate response with a valid client cookie.

Responses with BADCOOKIE Rcode Two of our conditions tested how a recursive resolver responds to a BADCOOKIE rcode. In one condition we still included the answer, but in the other, we did not. This imitates an authoritative server strictly requiring cookies (though a correctly behaving serving would provide a valid server cookie and accept it in future queries). For these conditions, we consider all 528,832 servers who successfully answered the normal condition regardless of cookie use.

For the BAD queries, 301,929 (57%) of IPs timed out and 206,577 (39%) returned an rcode of SERVFAIL. We observed similar values for BAD-ANSWER: 272,041 (51%) timed out and 236,401 (45%) returned SERVFAIL. We did observe an extra effort by recursive resolvers receiving either a BAD or a BAD-ANSWER response to get a valid response. More than half of IP addresses issued at least 19 queries in connection with either of these responses—as opposed to a median of 1 for NORMAL queries. Interestingly, 17,921 (3.4%) of recursive resolvers

[5] We did not rerun the initial collection as the process is resource intensive and takes multiple days. We are also less interested in servers lost due to churn as they are unlikely to be true open resolvers as opposed to misconfigurations.

that responded to our BAD-ANSWER query returned to us the answer that our servers had given to them, despite the `BADCOOKIE` rcode in the response from our authoritative servers. Of those that returned an answer, 14,350 (80%) also set the rcode to `SERVFAIL`. The lack of enforcement is accompanied by a lack of consensus on how unexpected responses should be handled.

5.2 Server Handling of Unexpected Client Behavior

Here we performed a short test to determine how DNS servers would respond to unexpected client behavior, with regard to the server cookie sent by the client. Specifically, we had our client send 5 queries that included the most recently received server cookie, 5 queries without a server cookie, and 5 queries with a fake server cookie. In each of these conditions, the client cookie was sent as normal. In the latter two cases, the specification provides three options for a server [11]. They may silently discard the request, respond with the `BADCOOKIE` error code, or respond normally as if no cookie option was present. We sent these queries to all Alexa IPs, TLD IPs, and recursive resolver IPs identified in section 3.1 that supported cookies.

For Alexa servers, we observed 41,083 IPs that responded to at least one normal query with a valid response and rcode of `NOERROR`. In our two other scenarios, nearly all of these IPs also had one or more standard responses: >99% for queries without cookies and with fake cookies. We observed 1 IP that used the `BADCOOKIE` rcode even when we sent the most recently received server cookie. We saw only 28 IPs use `BADCOOKIE` when we didn't send a cookie and 27 IPs when we sent a fake cookie.

For TLD servers, we initially observed 1,246 IPs that responded to at least one normal query with an rcode of `NOERROR`. All but 3 IPs returned an rcode of `NOERROR` in both the fake and missing cookie scenarios. These 3 IPs consistently returned an rcode of `BADCOOKIE` under these conditions, and all were authoritative for the `il` (Israel) TLD.

For recursive resolvers, we saw 137,896 IPs return an rcode of `NXDOMAIN` (we queried for a non-existent domain) for a normal query. Again we saw over 99% continue to behave normally when the server cookie was missing or fake. We measured 49 servers using `BADCOOKIE` for a missing cookie and 53 for a fake cookie (though 13 IPs sent `BADCOOKIE` incorrectly in the normal condition).

In summary, practically no server changes its behavior if it doesn't receive a server cookie or if it receives a fake one (even after the client previously sent valid cookies). While this behavior is consistent with the specification, it is the least restrictive approach. As a result, these servers can still potentially be used in reflection attacks because they will generate a full response regardless of the server cookie.

6 Discussion

We have now enumerated support for DNS Cookies and found that it is limited, both for clients and servers. We have also seen that few clients and servers

supporting cookies enforce them. This begs the question of what contribution, if any, DNS Cookies currently make. DNS Cookies are also in a difficult situation because they require wide deployment for enforcement to be enabled, but there may be little value in adopting them today. We now discuss the perceived current benefits of cookies and the path forward to wider adoption and enforcement.

6.1 Cookie Benefits Today

DNS Cookies have minimal value in their current state. We found that cookies are used by less than 30% of servers and 10% of recursive clients. This alone means that 70% of servers can be abused for reflection attacks and 90% of clients are not strongly protected from cache poisoning attacks (though other measures exist). Also noteworthy is the fact that 90% of clients are not sending server cookies (as a client cookie is a prerequisite).

Due to relatively low adoption rates, those that do support cookies are unable to enforce them since doing so would break compatibility with the majority of infrastructure. In our testing, we demonstrated our support for cookies in preliminary queries but still observed that only 20% of clients and less than 1% of servers changed their behavior if a cookie was missing or fake.

The only benefit we see today is that receiving a valid cookie acts as a reassurance that the other party's identity is correct. In real-world applications, this reassurance provides little value since it does not change an implementation's behavior: it would accept the message regardless of a cookie.

In summary, we do not see any benefits from DNS cookies, as they are used today. Cookies exist mostly in a dormant state, but if adoption significantly improves such that they can be enforced, they can become effective.

6.2 Path Forward for Cookies

The obvious next step for cookies is to increase adoption among clients and servers. However, there is somewhat little benefit to doing so today due to the lack of enforcement. Additionally, servers may not be concerned with identification (as they're only a passive entity in reflection attacks) and clients may feel protected from cache poisoning through other measures.

To incentivize adoption, strategies for partial enforcement should be explored. For example, clients and servers could begin enforcing cookies use for parties they previously observe using cookies. In our testing, we saw that 80% of clients and 99% of servers did not do this. Another enforcement implementation could involve a mechanism to advertise cookie support. This would allow other parties to verify that an IP intends to use cookies and then apply strict enforcement on a case-by-case basis. Neither of these enforcement strategies will overcome the lack of cookie adoption because enforcement can only ever be applied to the small percentage of clients and servers supporting cookies.

As a result, the main step for cookies is to continue to grow adoption numbers. As adoption grows, opportunistic or learned enforcement will become more

viable. Given the entrenchment of the DNS in internet infrastructure, it is unlikely that adoption will ever be universal, and as a result, strict enforcement may never be possible. Here we hope that strategic enforcement can be sufficient enough to deploy as a permanent strategy.

7 Ethical Considerations

All measurements and analyses performed in this paper were designed to be benign. Queries were sent at a low frequency, typically one per second, and never exceeded a volume of more than 20 queries per minute to a given IP address. Additionally, our probes were used solely to measure cookie usage and support. None of our probes were designed to exploit clients or servers.

8 Conclusion

In this paper, we present what is, to our knowledge, the first study of DNS Cookie usage. We find that cookie usage is limited, despite its standardization four years ago. We find that under 30% of IPs for the top 1 million Alexa domains and less than 20% of IPs for the TLDs supported cookies. We also observe that 17% of recursive resolvers support cookies as a "server", but only 9% do as a "client". We next analyzed a collection of server cookies and exposed potential issues, such as inconsistent clocks, which could potentially cause issues if cookies were enforced.

Finally, we experimented to see if any clients or servers enforced cookie usage. We observe that only 20% of clients and less than 1% of servers behave differently if an IP that previously supported cookies does not supply a cookie or replies with a fake cookie. This highlights that even those supporting cookies are not seeing any significant protection.

Overall, DNS Cookie adoption is limited, and there are few benefits for those using cookies. For cookies to leave their dormant state, higher adoption rates are necessary. From there, we believe that strategic enforcement may begin to produce real-world benefits.

Acknowledgments

We gratefully acknowledge the Comcast Innovation Fund for their support of the work that produced this material. We also thank the PAM 2021 reviewers and our shepherd for their helpful comments.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

1. dnscurve.org. https://dnscurve.org/ (2009)
2. Amazon: Alexa top sites (2020), https://aws.amazon.com/alexa-top-sites/
3. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: RFC 4033: DNS security introduction and requirements (March 2005)
4. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: RFC 4034: Resource records for the DNS security extensions (March 2005)
5. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: RFC 4035: Protocol modifications for the DNS security extensions (March 2005)
6. Bortzmeyer, S.: DNS query name minimisation to improve privacy (March 2016)
7. Böttger, T., Cuadrado, F., Antichi, G., Fernandes, E.L., Tyson, G., Castro, I., Uhlig, S.: An empirical study of the cost of DNS-over-HTTPS. Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC pp. 15–21 (2019). https://doi.org/10.1145/3355369.3355575
8. Damas, J., Graff, M., Vixie, P.: Extension mechanisms for DNS (EDNS(0)) (April 2013)
9. Deccio, C., Davis, J.: DNS privacy in practice and preparation. CoNEXT 2019 - Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies (2019). https://doi.org/10.1145/3359989.3365435
10. Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., Wessels, D.: Rfc 7766: Dns transport over tcp - implementation requirements (March 2016)
11. Eastland, D., Andrews, M.: RFC 7873: Domain name system (DNS) cookies (May 2016)
12. Hilton, S.: Dyn analysis summary of friday october 21 attack. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/ (2016)
13. Hoffman, P., McManus, P.: Rfc 8484: Dns queries over https (doh) (October 2018)
14. Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., Hoffman, P.: Rfc 7858: Specification for dns over transport layer security (tls) (May 2016)
15. Hubert, B., Mook, R.: RFC 5452: Measures for making dns more resilient against forged answers (January 2009)
16. Internet Assigned Numbers Authority: Root Files (2020), https://www.iana.org/domains/root/files
17. Lu, C., Liu, B., Li, Z., Hao, S., Duan, H., Zhang, M., Leng, C., Liu, Y., Zhang, Z., Wu, J.: An End-to-End , Large-Scale Measurement of DNS-over-Encryption : How Far Have We Come ? Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC pp. 22–35 (2019)
18. Mockapetris, P.: Rfc 1034: Domain names - concepts and facilities (November 1987)
19. Mockapetris, P.: Rfc 1035: Domain names - implementation and specification (November 1987)
20. Prince, M.: The DDoS that knocked spamhaus offline (and how we mitigated it). https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/ (2013)
21. Sury, O., Toorop, W., Eastland, D., Andrews, M.: Interoperable domain name system (DNS) server cookies (May 2020)