A rather common form of optimisation of DNS name resolution is for authoritative name servers to base their replies on the source IP address of the resolver in order to improve the performance of services in that namespace. In this paper we explore a different approach, where the querying resolver explicitly includes a potentially generic hint in DNS queries. This allows nameservers to implement optimisations that take into account hints supplied by the application resolving a DNS query. Conceptually this is similar to the approach taken by Content Distribution Networks, and widely distributed services that use DNS to distribute their load based on source locality. Our solution differs in purpose and approach, since it explores user handles from the services that own the nameservers as resolution hints, and considers the resolver as explicitly involved in this process.

I. INTRODUCTION

A common discussion topic concerning DNS name resolution is whether authoritative name servers should supply different replies based on the source of a DNS query. No general consensus exists on this question; nevertheless several authoritative name servers are known[1] to yield different replies based on the topological location of the resolver, derived from the source IP address of the query.

The reason authoritative name servers assume this behaviour is to transparently support service optimisations. This allows services to explore models based on Content Distribution Networks, where the resolver gets an answer that targets the closest content server, or content servers with localised versions of content. Unfortunately since two resolvers at different locations may retrieve distinct results, and the process is transparent to the querying resolvers, DNS ends up losing its global consistency.

Regardless of this shortcoming it would seem this use of DNS is here to stay. Our objective is to explore a different use case where instead of having the DNS nameserver unilaterally triggering these optimisations, the resolver will explicitly insert information, in the form of a resolution hint, within DNS queries in order to trigger this behaviour.

Our motivation to pursue this venue is twofold: First, this would be an opt-in feature; the resolver would be explicitly aware that the results being returned might not be globally valid, and would not insert them into its name caches, at least without understanding the potential hazards. Second, this will empower the nameserver with the capability to make better decisions, that do not explore the locality of the resolver, but instead use service specific hints supplied by the resolver.

What we propose in this paper is a method for a resolver to embed service specific information in a DNS query, whose purpose is to signal a specific authoritative nameserver that the answer should be tailored based on the embedded hint. Such mechanism could then be used to explore service specific customisation.

Furthermore we explore one particular case, where this hint refers to the identity of the user behind the resolver, and discuss the added advantages and implications.

The remainder of this paper is divided as follows: section II describes the related work, section III describes the fundamental aspects of our solution, section IV explores the identity case and describes our implementation for the case of a specific authentication protocol, in section V we discuss the proposed solution and conclusions are presented in section VI.

II. RELATED WORK

The use of DNS as a directory service, for load balancing or localised service decisions has become a common unintended use for the DNS infrastructure. This kind of use of DNS is heavily criticised [2], because it provides differentiated responses based on the request IP source address. This behaviour hinders the use of DNS caching because it provides responses that target a subset of the possible resolvers.

Furthermore the use of the source IP address of a query as means to determine the location of the source resolver, is not necessarily accurate given the use of recursive resolvers.

However the line of reasoning that supports the introduction of this kind of service policies in DNS continues to be used [1]. A recent DNS extension draft [3], proposes that recursive nameservers should embed IP address information of the original resolver within the query that is sent to the authoritative name server. The draft in question does address several concerns with this kind of usage of DNS, it defines appropriate caching for intermediate resolvers and provides authoritative nameservers with more accurate location for the resolver. What it doesn’t do, is justify why would an intermediate DNS resolver forfeit user information to a service, neither does it explore this as an opt-in mechanism for users1.

1The draft does offer the means for a resolver at the terminal to opt out, but requires the terminal to invalidate the option in all outgoing queries.
The method proposed in this paper to embed resolver information within queries, mirrors that of RFC5001 [4], where a nameserver places an identifier in OPT RR records for responses. However our purpose it to have the resolver placing information in DNS queries, that may function as a resolution hint for services.

III. SERVICE SPECIFIC DNS QUERIES

As was mentioned earlier, services benefit from introducing some extra bits of logic within their DNS nameservers because they get to tune performance based on network locality, and target localised user content. This kind of procedures is a form of location aware resolution that attempts to improve the user/service experience based on the location of the resolver. However these mechanisms completely ignore the preferences of the user behind the resolver and in some cases the service might want improved performance based on properties that are not related to the locality of the client resolver.

Our intent is introduce a generic form of resolution hints, where the resolver places additional information within DNS queries. The purpose of such hints may vary from service to service.

For localisation purposes, the user application can place an hint such as en_GB to indicate language preferences. The nameserver will then provide an A record pointing to a server with localised content.

To provide context information, for example placing a indicator of the type of connection the user terminal was using. Giving preferences to service instances that use higher, or if that is the case, lower bandwidth.

For service performance issues, one can consider a scenario where, the bottleneck is not between the user terminal and HTTP server but rather between the server and a user profile database. In such scenario it would be preferable to consider the locality of the database rather than the user terminal. Large social networking services, and federated networks of services, are good examples of such cases, because services have to poll frequently from a user profile information and most operations involve user information to filter results. In such cases it is often preferable to choose an HTTP server co-located with the user database that holds user information.

One particular case of hint, which will be our focus over the remainder of this paper, concerns the case where instead of placing an attribute in the query we place an identifier that ties a user to the service. In our view, when making a DNS query for a service, we could embed in the query our service username or email address. The authoritative nameserver could then provide an answer based on our user profile information. Pertaining the example above, about localised content, the nameserver would now fetch user localisation preferences from his profile, rather than extracting them from the DNS query.

What we propose ourselves to implement is a solution that allows an "intelligent" DNS resolver to signal to a nameserver that it wants results tailored to a certain hint. The hint we employ here is service specific (a handle, username, email), and reflects an existing registration with the service that manages a given domain. The core principles for our solution are the following:

1) These queries are service specific, a consumer and a service are negotiating differentiated conditions.
2) These hints are not always enabled, the application behind the resolver must explicitly ask for this kind of resolution.
3) The originating resolver is explicitly involved. It is the resolver that triggers this form of resolution, and it is aware these results are not globally valid.
4) No intermediate resolvers or caching are used, these results are meant for one specific resolver.

We choose to focus on the case, where the information hint is in fact a user identifier that defines a registration relation between a user and a service. This is the case that maximises the opportunity for custom resolution, because it attempts to link service customisation in DNS, to the user identity (and profile information) at the service.

All this implies the resolver at the user’s terminal will have to be, fully recursive and capable of keeping additional state information. We envision this solution as a resolver library that can be part of applications that require this kind of functionality.

A. Resolution Hints

To introduce our resolution hint we alter DNS queries and responses (Figure 1) to the authoritative nameserver by including an EDNS0 [5] OPT record. The record includes an additional field called Resolver Hint where the resolver encodes resolution hints. The meaning of the field is service specific, since we focus on user identification it can be a user handle such as a username or email address encoded as a TXT records [6], as long as it allows the authoritative name server to uniquely identify the issuer of the query.

If the nameserver does not support these extensions, the query will simply fail, the nameserver will yield a DNS error reply, as defined in RFC2671. The resolver can then retry without a Resolver Hint, for a regular result.

Fig. 1. Additional Options in DNS messages, grey fields are EDNS0 headers
Terminal

means to authenticate the request, based on the Resolver Hint by TSIG [7]. Instead we would like to introduce an alternative nameserver, but this kind of authentication is already covered key between the resolver and the service that manages the hint to authenticate the resolver, for example based on a shared within the DNS query we can also authenticate the query. this we consider that based on the user handle embedded other terminals. This implies that the enquiring resolver must particular resolver, which means that it can’t be shared with other terminals. This implies that the enquiring resolver must behave as a recursive resolver, rather than entrust the outcome to another recursive resolver.

But even if the resolver will deal directly with the authoritative name server, since DNS offers no means to ensure confidentiality of query responses, an eavesdropper could see the content of the reply. Given this issue, the service whose authoritative name server issued the response should not assume the response to be a secret. We do however assume the authentication channel to be secure.

One final consideration about the inclusion of user identifiers within DNS queries, is that the presence of such identifiers within clear text DNS messages will disclose user information to eavesdropping third parties and could be used by eavesdroppers to track user presence at the access network. This can be properly addressed using pseudonyms instead of the actual username of the user behind the resolver, a pseudonym that the nameserver would recognise as equivalent.

IV. IMPLEMENTATION

We could of course insert enough information within the hint to authenticate the resolver, for example based on a shared key between the resolver and the service that manages the nameserver, but this kind of authentication is already covered by TSIG [7]. Instead we would like to introduce an alternative means to authenticate the request, based on the Resolver Hint and existing push based authentication protocols.

The process of actually verifying that the identifier attached to the query is owned by who issued the query, is handled aside from the regular DNS messages. It is up to the authoritative nameserver to trigger an out of band authentication, sending the query information to the holder of the Resolver Hint for confirmation.

The actual mechanism that is proposed here is not tied to any specific authentication protocol. The service behind the authoritative name server has to determine which protocol and settings to use, likely based on existing settings already at the authoritative nameserver. A specific example is described when we discuss our implementation in section IV.

We assume that both the service and the user behind the resolver share a preexisting registration and that the service can authenticate the request using a push based authentication protocol. As such we delegate the actual authentication process on external protocols, because this frees us from further burdening DNS messages, and defers to proper authentication protocols.

C. Privacy Considerations

Given that one of the key use cases for this solution is to authenticate the resolver of a DNS query, we must forfeit the usual stub resolver behaviour. In case of success the result supplied by the name server will be tailored to that particular resolver, which means that it can’t be shared with other terminals. This implies that the enquiring resolver must behave as a recursive resolver, rather than entrust the outcome to another recursive resolver.

But even if the resolver will deal directly with the authoritative name server, since DNS offers no means to ensure confidentiality of query responses, an eavesdropper could see the content of the reply. Given this issue, the service whose authoritative name server issued the response should not assume the response to be a secret. We do however assume the authentication channel to be secure.

One final consideration about the inclusion of user identifiers within DNS queries, is that the presence of such identifiers within clear text DNS messages will disclose user information to eavesdropping third parties and could be used by eavesdroppers to track user presence at the access network. This can be properly addressed using pseudonyms instead of the actual username of the user behind the resolver, a pseudonym that the nameserver would recognise as equivalent.

IV. IMPLEMENTATION

The focus of our implementation is the identity use case, and for that end our implementation is split in three parts: a DNS resolver library that can trigger authenticated requests when called from a specific API; second a modified authoritative DNS name server that processes the queries and triggers the authentication process, and finally an authentication agent that handles pushed authentication requests. Based on these three components we implemented a complete example by using an XMPP side channel to authenticate DNS queries issued to our nameserver.

A. Resolver

Our resolver implementation is based the resolution libraries provided by the Unbound project [8]. It provides a specialised API, similar to the POSIX gethostbyname() APIs. The main difference is that this new API takes two additional parameters, the resolver hint represented as a string, and the authentication options as a 32bit field. The resolver encodes this information in outgoing DNS queries in an OPT record.

The resolver itself does not employ caching for these special queries, since each query is specific to the queried name and information hints. We do however provide a per application cache as part of the API.

B. NameServer

The authoritative nameserver implementation that was produced, is based on an existing open source implementation [9], modified in order to store user specific records. This means that in addition to the regular records, it can hold alternative query answers on a per user basis.

The implementation behaves like a regular authoritative name server, except when it receives a message with a Resolver Hint. In which case the nameserver checks the Options field and matches it against a table of authentication methods. It then feeds the arguments from the query as parameters for the authentication method and waits for its completion.

When yielding the query result back to the resolver, the nameserver may signal error conditions in the Options field,
error conditions include an authentication failure, or an invalid value for the **Options** and **Resolver Hint** fields.

**C. Authentication Agent**

For demonstration purposes we authenticate queries based on XMPP XEP-0070 [10]. We choose this particular protocol because it is a simple example of a push-based authentication protocol that can be extended for our purposes, other examples could be considered based on different protocols.

The XEP-0070 specifies a XMPP extension to authenticate HTTP requests using the XMPP protocol. The protocol queries the owners of a certain JID for confirmation of the validity of an HTTP request, based on its target and transaction ID. It allows an HTTP server to accept a user handle in the form of `user@domain`, the server uses that handle to reach its owner using a XMPP transport to confirm that it is indeed responsible for that request.

In this example the hint placed in DNS queries is a XMPP JID in the form of `user@domain`. Upon receiving a query with an attached **Resolver Hint** the nameserver will issue a confirmation request to an XMPP server (Figure 3), that in turn will send it to owner of the handle.

A XMPP agent collocated with the resolver will then receive the request, verify the validity of the query against the state of the resolver and then respond accordingly. Upon receiving the response from XMPP, the nameserver will finish the DNS query by responding to the resolver.

**V. Discussion**

The mechanism we propose allows a DNS resolver (usually at a user terminal) to signal to an authoritative name server that it is interested in custom answers, by placing resolution hints within DNS queries. This requires the resolver to act as a recursive resolver, communicating directly with the authoritative name server, and circumventing any potential intermediate resolvers (and caches). Since we now tie DNS request handling to an authentication protocol, the answer to a query will have to wait for the authentication to be complete. This can introduce a significant increase in delay for DNS queries, that is dependent on the authentication protocol that is used.

DNS does not specify a strict timeout value for resolvers, instead it leaves to resolver implementations the task of defining such value [11], [12], and implement a back-off mechanism. We consider no special mechanism to address this, other than raising the initial query timeout for the resolver, however even if the initial timeout value is too short the resolver should back off properly [12] without changes. All this comes at the cost of additional load on the authoritative name server, since we’ve effectively split the load between the resolver and the authoritative nameserver.

We see this as an appropriate compromise, since we don’t consider this to be on by default and we only target this behaviour at enabled nameservers that have explicitly enabled this feature for some services.

The inclusion of a user handle hint in DNS queries implies that the handle, a registration relation with the service, already exists along with proper setup for the authentication protocol. This limits the applicability of our solution to cases where the resolver, or rather the application behind it, knows beforehand which user handle to employ. But this is actually the case for several common examples: for example automated email agents, or webservice consumers (such as Twitter clients), that periodically poll a service for user specific content. These are exactly the kind of services that we foresee as being able to leverage this, because they are heavily focused on user identity and require pre-configuration at the user’s terminal.

**A. Intermediary resolvers**

One implicit assumption for our proposal, is that our specialised resolver is placed within the user’s terminal and attends a specific API called by service applications.

If the intended objective were to take a similar approach as the one seen in [3], our modified resolver could be running as an intermediate resolver and the terminal would use a regular stub resolver. However this would result in the same pitfalls described earlier, since the resolver would be unaware of this process.

Still our modifications are flexible enough to allow this kind of use, and in some cases this approach is justifiable. For example if a recursive nameserver in the access network and the name being resolved are part of the same administrative domain (the same network for example), the local resolver may use this method to convey a hint to the authoritative nameserver.

**B. Authentication protocols**

Our implementation and description was focused on a scenario where push based authentication methods were employed. Other alternative would be to include enough information in the hint field to authenticate the query, for example signing the field with a key shared between the terminal application and the service.

However we intended to have a solution that was independent from the actual authentication protocol, and could leverage protocols already used by the service supporting the authoritative name server. Additionally DNS query authentication solutions based on shared keys could be derived from existing DNS extensions, such as TSIG [7].

**C. Service Customisations**

We don’t enforce any format for the resolution hints within queries, since services often have their own namespaces of user identifiers and attributes. Our initial intent was to place generic resolution hints within DNS messages, we focus on the cases were the hint refers to the user identity because this allows for intensive customisation, and also because it is the most reasonable solution for scalability purposes.

Naturally we cannot hope to encode an infinite amount of attributes within a DNS query, or the system would not scale.

---

2The Extensible Messaging and Presence Protocol is the base of the Jabber instant messaging, and the XEP-0070 extension is widely supported by clients and development libraries.
If the hint uniquely identifies the user, this can be explored further. The nameserver can access user profile information, and use it instead, to determine user preferences.

The use of the user identity as a hint allows some interesting customisation options. Even if we don’t authenticate the DNS queries (section IV), we can consider the value of the Resolver Hint as a hint to pre-load user information from the database in preparation for a user connection over HTTP.

If authentication of queries is enabled then prioritisation of services can also be implemented, by returning different hosts for different users. The same DNS lookup for service.com could return a CNAME response of gold.service.com or silver.service.com depending on the user’s identity.

VI. CONCLUSIONS

In this paper we presented a solution to include information, in the form of a resolution hint, within DNS queries. The proposed solution allows a nameserver to yield query results that are tailored to a specific resolver, enabling the services that are in control of such nameservers the ability to offer DNS results based on the identity of its user. Since we treat this as a process of hinting a specific service, all DNS caching based on intermediate resolvers is forfeit, but we also open the possibility of leveraging the user’s identity as means for service optimisations.

We produced an implementation of a DNS resolver, and DNS nameserver, that use an XMPP channel for authentication of DNS queries allowing a DNS nameserver to authenticate queries and respond based on the outcome of the authentication process. The method we use, and the produced implementation, is flexible enough that it could use other authentication protocols as long as they can derive a push based authentication out of the user identifier. This allows services to signal user identification, within DNS messages to its nameservers, but requires the terminal to run our resolver library and support any push based authentication protocol that is used.

Even if this is not a replacement for similar custom DNS resolution techniques, it does allow for interesting possibilities for applications that are already expected to require support at the terminal, and does so with explicit involvement of the terminal.

REFERENCES


Fig. 3. Authenticating DNS queries using a XMPP JID <user@domain>