Securing DNS Communication for Client Systems with dynamic DNS Filtering

Philipp Kalytta
Institut für Nachrichtentechnik
Technische Hochschule Köln
Cologne, Germany
pkalytta@th-koeln.de

***Abstract*—Dynamic DNS filtering applications use the DNS system to filter out DNS answers to clients, that query for malicious hosts or content. This traffic between the client and local resolver as well as to upstream nameservers is commonly not protected against attackers. This paper provides an approach to securing two commonly used filtering DNS resolvers between the client and itself and on the upstream side and compares them to Unbound, a validating, recursive, caching DNS resolver and its features.**

Keywords—DNS, DNS-over-HTTPs, DNS-over-TLS, DNSSEC, DNS filtering, confidentiality, integrity protection, Pi-hole, AdGuard, Unbound

I. Introduction

The Domain Name System, described in RFC 1034 [1] and specified in RFC 1035 [2] provides the Internet standard mechanism for name to IP address resolution, which was revised and added upon multiple times. Relevant RFCs in the context of this paper are [1], [2], [3] regarding DNS, [4], [5], [6] and [7] regarding DNSSEC and [8], [9] regarding encryption for DNS. DNS is an integral part of digital communication, but the standard protocol provides no security: It either uses unencrypted UDP over port 53 or unencrypted TCP over port 53 to resolve names. Also, the requested names in a query to a DNS server, could itself be part of a malicious intent: As the user has no control over the target domain name and what domains the hosts queries for (at least in a common setup), there is also no control for the user, to decide what content to get – it is entirely dependent on the remote nameserver. This, in combination with a global rise in internet advertisements [10] has led to the development of DNS blocking solutions, that effectively stop the local host from resolving the domain names of advertisement services or malicious content by using curated lists of blocked domains. These services, when deployed, mostly use the standard DNS over Port 53, either by recursively resolving domains by themselves [11], or by using one or more upstream (recursive) resolver servers (i.e. Google and Cloudflare provide such servers [12] [13]), that will then directly answer the query to the local resolver/DNS software. All this communication is not secure by default. This paper presents different approaches to securing two commonly used DNS filtering applications and compares their filtering capabilities to the Unbound [14] DNS resolver software. Local resolver in the context of this paper is always the network-local server that receives and answers queries from the network clients (i.e. The Pi-hole server, AdGuard server or Unbound server).

II. Objective

A. How DNS Filtering works

DNS filtering solutions mainly rely on intersecting the DNS traffic of clients: By configuring the DNS filter software to act as the DNS resolver for a client, the client will send its DNS queries to the resolver. This resolver will then typically compare the requested domain name to internal blocking lists or regular expressions that define filtering parameters. If the domain does not match with these, the domain will be resolved by the software normally. If it matches the list or a regular expression, the resolver will not continue to resolve the domain, but will instead send an answer to the client, that contains bogus data (i.e., NXDOMAIN or a wildcard IP like 0.0.0.0): The client will therefore think, that the domain does not exist or at least will not be able to connect to the hosts behind the domain name. Therefore, the request is effectively filtered out.

This paper focuses on the two most popular (by GitHub Stars [15] [16]) open-source self-hostable DNS blocking resolvers: Pi-hole and AdGuard Home. There exist other self-hosting solutions like eBlocker and a number of services, directly provided by large DNS resolvers, like Cloudflares Family Filter [17].

DNS filters have the same common problem as every other conventional DNS software: By default [18] [19], they do not use or enforce extensions to the domain name system, that provide confidentiality [8] [9] and/or integrity [4]. Neither towards the upstream servers, nor towards clients. The goal of this paper is to show these techniques can be deployed to DNS filtering software and how this prevent or mitigates threats against DNS communication. The special focus is on how these protocols interact with each other in blocking and non-blocking scenarios and to elaborate on how these solutions can work together.

B. Insecure DNS Filtering Solutions

By default, only AdGuard Home achieves confidentiality for the upstream resolving queries via DoH to Quad9, but it does not verify the integrity via DNSSEC and does not provide a secure way of communication for its own clients. Pi-holes initial security is even worse: It does not even use encryption for an upstream connection and also does not provide it for clients, as well as no DNSSEC validation (Integrity-protection can also be a problem, if enabled: See Chapter V). As browsers are slowly deploying their own secure DNS services to clients [20], it becomes less desirable to use a local blocking DNS resolver, if it does not support at least the same security principles. These insecure filtering solutions lose their characteristic of increasing security by blocking if they do not provide the same security regarding confidentiality and integrity and protect against the common threats against DNS.

C. Existing Threats against DNS

RFC 3833 [21] lists known threats to the Domain Name System, of which some threats/special cases are presented here again as information. Different sources also name

different lists of common threats [22] [23] [24] which overlap mostly on these threats:

1) DNS Cache Poisoning

By sending wrong (spoofed) answers to the local DNS resolver – either by sending them faster to the local resolver than the original server or by sending them from the responsible but hacked nameserver, an attacker can incorporate these resource records in the cache of the local resolver. Because DNS is not integrity protected, the local nameserver has no way of knowing if the DNS data it received is correct and must assume it is the right data. The original DNS protocol stack contains some implicit mitigation for this case (The Transaction ID [25]), but not in a cryptographically secure way and it is known to be insecure [25].

2) Flooding Attack

There exist different types of flooding attacks against DNS resolvers. As name resolution can be performed via UDP as well as via TCP, both variants are attack surfaces for a flooding attack. The goal of this attack is to overload the server or interrupt the communication by exhausting the system resources and available connection pool. There are variations like the NXDOMAIN attack, which tries to overload the resolver by asking for nonexistent domains, or the random subdomain attack – or water torture attack – that floods the server with random subdomains to try to overload it.

3) DNS Rebinding

DNS Rebinding enables the attacker to target hosts inside the local network of the victim. It effectively circumvents the same-origin policy most browsers use to prevent JavaScript or other locally executed code to access other systems that that it originated from. After loading the code from the attacker's server into the victim's browser, the attacker changes the resource record associated with the domain, this can be done very fast by setting a low TTL for the resource record. The attacker changes it to a local IP address, this way, it is possible for the script to attack a local host from the victim's machine. This is simplified by the fact, that most (home) networks have the same layout (i.e., the router is having the first IP address of the segment). This is often mitigated by denying responses from nameservers on the internet that contain local IP addresses or by setting a minimum TTL for all query answers, which slows these attacks down.

4) Phantom Domain Attack

Phantom domains are domains, where the responsible nameserver either does not answer or answers with a large delay (This can be induced by the attacker). Querying for those domains has the goal to make the local resolver wait for the responses as long as possible to consume open connections. This can be mitigated by setting a short max-wait time for outgoing queries for the local resolver, assuming that most nameservers will answer very fast (which must not be the case).

5) Man-in-the-Middle Attack

An attacker that performs a Man-in-the-Middle (MITM) attack can intercept all DNS traffic and can change or deny responses or queries at will. As DNS is not integrity or confidentiality-protected, this does only require the MITM to route the DNS traffic through itself, which, for example is possible by sending out rouge DHCP answers to clients in the same network containing wrong DNS data or by using the NDP for IPv6 [26].

III. Solution Approach

The process for improving the security of DNS communication for the previously named DNS filtering applications on the downstream to clients as well as on the upstream was divided into the following parts:

1) Identification of existing threats: This was covered in the previous chapter.

2) Composition of possible countermeasures: A list of measures that improve either in one or more parts of the three primary focus points of information security: Confidentialty, integrity and availablilty of data.

3) Technical outline of the interaction of the measures: This defines how the different aspects of the measures interact and they complement each other to improve security.

4) Description and setup of a test environment: Implementation of a testing setup enables us to test for improved security and enables the testing of features like interaction (i.e., domain blocking) via API.

A. Countermeasures

1) DNSSEC

The Domain Name System Security Extensions (DNSSEC) which are described in RFC 4033 [4] are adding integrity of data and data origin authentication to the DNS by using digital signatures over a set of resource records. It also adds authenticated denial of existence of DNS records. This protects against different type of attacks, i.e., a MITM cannot generate a correctly authenticated DNSSEC secured answer to a query for a domain name that does not exist. Its nonexistence is provable by the DNSSEC records in the parent zone. In fact, it protects the DNS against most of the threats presented in RFC3833 [21] [27]. It does not introduce transaction security in the sense of confidentiality. Most stub resolvers do not enforce DNSSEC validation but rely on the local recursive resolver to validate. This requires, to prevent manipulation of the data in transit between the local resolver and the client (stub), a form of secure channel (encryption) between resolver and client. It also cannot protect against denial of service (DoS) attacks but introduces some additional DoS possibilities [28]. Also, a new attack, that was previously only possible via zone-transfers, is possible: Zone enumeration. This is targeted against authoritative zones and nameservers and not further discussed here.

2) DNS-over-HTTPS and DNS-over-TLS

The internet standards DNS-over-HTTPS (DoH) described in RFC 8484 and DNS-over-TLS (DoT) which is described in RFC 7858 provide the ability to eliminate eavesdropping on and tampering with data in transit by providing transport encryption [8] [9]. A client can either try to opportunistically learn about a privacy enabled DNS server by attempting DNS over TLS on port 853 or it can rely on an out-of-band configured privacy profile, where the server and its key is known to the client, this provides a stronger trust between server and client – The RFC for DoH explicitly excludes the opportunistic approach for DoH capable clients, this is only described for DoT. Deploying a DoT or DoH URI to the clients to use for requests, always requires either trying the known DNS server for support of DoT or manual or automatic configuration via out-of-band methods (i.e., by providing the URI via DHCP). Automatic configuration entails other security risks (i.e., rouge DHCP server).

3) DNS Rebinding Protection Options

DNS rebinding can be prevented by configuring the DNS server to error on answers coming from the internet containing private IP addresses. This can affect some DNS enterprise deployments that rely on name resolution for internal IP addresses for their own domain by nameservers that are reachable from the internet. There exist config options to circumvent this [29] [30]. Also, the named applications include a local cache. The minimum TTL for the cache can be configured to never fall below a given limit, which slows a rebinding attack down.

4) Rate-Limting and Timeouts

Overloading the server can be prevented by limiting two factors: The number of queries that the server accepts in a timespan and the time the server waits for a response from upstream servers. Common DNS server software allows for rate limiting and setting timeouts to help with overloading attacks [31].

B. Technical Outline

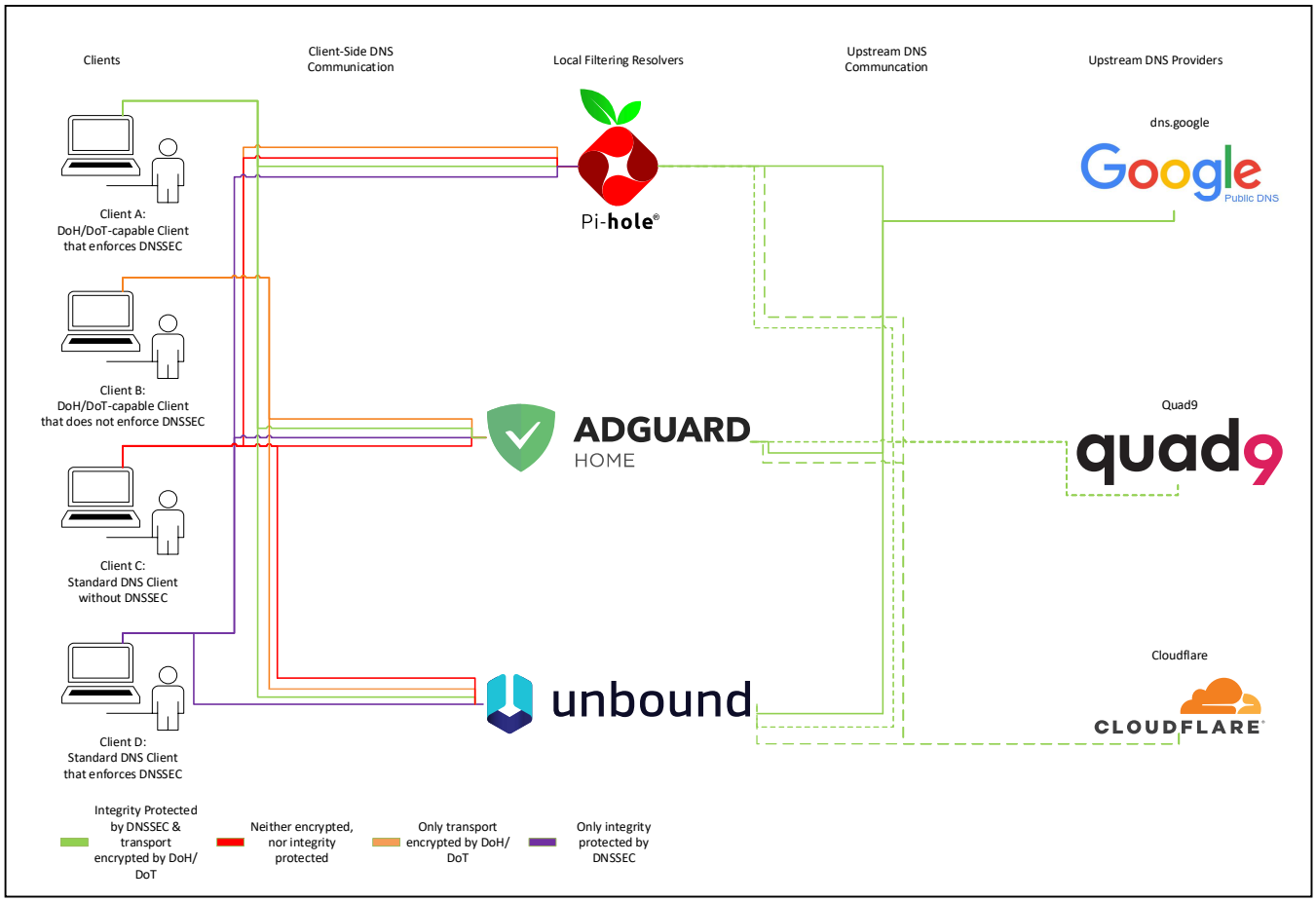
To achieve integrity- and confidentiality-protected DNS communication between a local client and the local (recursive) DNS resolver, the following technical criteria must be met:

- The local resolver must support DoH or DoT, preferably both, to achieve transport encryption. There might be multiple steps to take to enable this for the client as well as for the upstream resolving process (When the local resolver is *recursive*, this might even be impossible, as there is no good way for a nameserver to indicate that it does support DoH or DoT).
- The local recursive resolver must detect, if upstream servers support DNSSEC (so sending a DO flag and expecting an AD flag from the upstream server) should validate resource records that it receives. This ensures, that at least for zones that support DNSSEC, the integrity can be proven. As the local resolver has blocking capabilities, it might block some of the DNSSEC signed records. If the client also enforces validation of the records it receives, thus validating against the DNSSEC chain, the client would be able to detect the blocking of the correct query answer. Clients should be configured to by default not strictly validate answers received from their local recursive resolver and must trust the resolver that it validates the incoming answers for them.

To further reduce the attack possibilities, countermeasures like the ones described in the previous chapter, that try to prevent DNS rebinding and Denial of Service should be able to be implemented by configuring the software. If there are no obvious configuration options, it can be tested if this is already part of the default behaviour.

C. Lab Environment and Software

As mentioned before, the selected software applications are AdGuard Home, Pi-hole and for comparison purposes and to analyze the integration possibilities of for DNS blocking into by default non-blocking resolvers: Unbound. All can act as a local caching resolver, and all can be configured to be DNSSEC aware and validating. AdGuard by default supports upstream DoT and is configured with Quad9 as upstream DNS recursive resolver. It can only act as a forward resolver and expects its own upstream recursive resolver. This is also true for Pi-hole: The documentation also mentions Unbound



explicitly to function as upstream recursive resolver [32]. Unbound can function either as forwarding resolver or can function as a recursive resolver.

By default, AdGuard Home supports DoH and DoT on the client facing side, if configured with a valid X.509 certificate. It then can be activated in the web interface. Pi-hole does not support this configuration out-of-the-box. In the Lab environment, this was achieved by installing BIND9 on the same host as the Pi-hole software and setting the local Pi-hole DNS resolver as upstream resolver for BIND9. BIND9 was then configured to serve as DoT and DoH resolver for the client. As this setup shared the same IP addresses for the Pi-hole software and BIND9 software (same host), it is indistinguishable for the clients and acted as one DNS server supporting all protocols from the outside. This can also be achieved with Unbound instead of BIND9. After this addition to the Pi-hole host, both hosts (Pi-hole and AdGuard) allowed for confidentiality and integrity protected communication to the clients. To achieve upstream confidentiality for Pi-hole, which is also not possible by default, cloudflared, a daemon that supports receiving and forwarding DNS queries via secure means to upstream servers, was locally installed on the Pi-hole host. This complemented the system to be secure client- and upstream-side. Unbound, as the third solution, can be directly configured with a X.509 certificate to allow for secure client configuration, as well as can be configured to use either TLS or HTTPS for upstream DNS communication, then, disabling its recursive features.

1) Scenarios and combinations considered

The following scenarios were configured and tested in practice in the lab environment and compared regarding expense, complexity, problems, common risks, and common threats (See Chapter IV.):

TABLE I. LAB ENVIROMENT TESTING SCENARIOS

Scenario	Combinations			
	Resolver Software	***Filtering***	***DNSSEC***	***DoT/DoH***
Pi-hole plain DNS	Pi-hole	Enabled and tested for specific domains[a]	Not requested	Not used
Pi-hole plain DNS with DNSSEC	Pi-hole	Same as above	Requested and validated	Not used
Pi-hole DoH with DNSSEC	Pi-hole	Same as above	Requested and validated	DoH used
Pi-hole DoH without DNSSEC	Pi-hole	Same as above	Not requested	DoH used
Pi-hole DoT with DNSSEC	Pi-hole	Same as above	Requested and validated	DoT used
Pi-hole DoT without DNSSEC	Pi-hole	Same as above	Not requested	DoT used
AdGuard plain DNS	AdGuard	Same as above	Not requested	Not used
AdGuard plain DNS with DNSSEC	AdGuard	Same as above	Requested and validated	Not used
AdGuard DoH with DNSSEC	AdGuard	Same as above	Requested and validated	DoH used
AdGuard DoH without DNSSEC	AdGuard	Same as above	Not requested	DoH used
AdGuard DoT with DNSSEC	AdGuard	Same as above	Requested and validated	DoT used
AdGuard DoT without DNSSEC	AdGuard	Same as above	Not requested	DoT used
Unbound plain DNS	Unbound	Same as above	Not requested	Not used
Unbound plain DNS with DNSSEC	Unbound	Same as above	Requested and validated	Not used
Unbound DoH with DNSSEC	Unbound	Same as above	Requested and validated	DoH used
Unbound DoH without DNSSEC	Unbound	Same as above	Not requested	DoH used
Unbound DoT with DNSSEC	Unbound	Same as above	Requested and validated	DoT used
Unbound DoT without DNSSEC	Unbound	Same as above	Not requested	DoT used

[a] There were multiple tests conducted: One domain that was configured to be blocked but not DNSSEC signed, one that was blocked but comes from a signed zone and one domain that was configured to not be blocked for comparison.

Fig. 1. Combinations of different test parameters for the different capabilities of the used DNS software that were practically evaluated on *client facing side* of the local resolvers.

Upstream on all three resolvers was configured to use DoH to either Cloudflares DNS resolver, Google or Quad9.

2) APIs for Dynamic DNS Filtering

Pi-hole as well as AdGuard support two types of static DNS filtering: They can use a list of domains the user can input themselves (user-defined), that can use regular expressions to detect domain names and can pull block lists from the internet (which are curated externally) which are mostly in the format of a hosts-file [33]. Pi-hole and AdGuard both have a web interface that allows for manual configuration of theses static lists. The internet-provided lists are typically updated in regular time intervals (i.e., once per day). AdGuard additionally maintains lists for common services (i.e., Netflix, Facebook, Instagram) to specifically block these services. These three types do not directly allow for dynamic detection of malicious or unwanted domains that are not in these curated lists.

Achieving dynamic DNS blocking requires at least a basic API that allows for two things: Getting informed of the domains that are requested by clients and adding and removing domains from the blocklist. Ideally, it is possible to decide if a domain should be blocked on the first query for the given domain name, so that the answer from the blocking local resolver can be directly changed if needed. Pi-hole does only

provide a Telnet based API for its internal FTLDNS service, that allows only receiving statistics [34] – this concludes that Pi-hole is not feasible for directly supporting dynamic DNS blocking without changes to the software.

AdGuard provides an OpenAPI specified HTTP-based API that is documented [35] and provides the log-API for getting information about queried domain names. This does not enable AdGuard to block initial DNS request, but only enables the API user to retrospectively act by blocking the domain name via the filtering-API. For comparison we can look at Unbound and BIND9, which are non-blocking resolvers by default, but their API can be used to enable dynamic blocking:

With libunbound, the Unbound server provides an API for manipulating incoming and outgoing DNS requests and responses [36]. The API even has a Python interface [37], which is a thin wrapper around the C API originally provided. There exist examples [38] on how to locally manipulate zone entries, which can be used for blocking-in-place, by using the API to provide a blocking-capable DNS service that checks incoming requests and decides if they will be blocked or not.

For BIND9, there is the possibility to use Dynamically Loadable Zones (DLZ) or DynDBs [39]. DLZ extend BIND9 to load domain data from an external database, this interface can be exploited to – instead of linking to a database – link to a dynamic decision algorithm that decides on the blocking. There exists documentation for that API, but its dated [40]. The DLZ API is also unable to handle DNSSEC data, which is possible with DynDBs: RedHat created an LDAP back end for the DynDB interface in BIND9 which allows for dynamic domain names via LDAP [41]. The ISC also provides a test implementation of a driver for DynDB via their GitHub repository [42].

IV. Results

Securing the DNS filtering/blocking software applications each requires solution approaches that differ in expense, complexity and allow for different levels of security. Not in all configurations, all threats can be mitigated. It shows that secure by default and security by design is not yet possible in the DNS ecosystem, especially when using DNS blocking software.

A. Expense And Complexity

Installation of all three applications is automatic, the initial configuration works out of the box for non-secure communication. Enabling DNSSEC validation for all three can be easily done by enabling the configuration option. This only requires an upstream resolver, that also supports DNSSEC, which can be chosen freely by the user (i.e., chose the one you trust the most). AdGuard reduces the expense and complexity regarding the deployment of DoH and DoT by allowing the upload of a X.509 certificate to the web interface, which then in turn automatically activates DoH and DoT to clients. For upstream, it already comes enabled with DoH and DoT support. It even experimentally supports DNS-over-QUIC. Pi-hole neither supports upstream DoH/DoT nor on the client facing side. To enable it for clients, an additional local resolver like BIND9 or Unbound must be set up on the same host, to accept DoH/DoT connections from clients and forward them to the local Pi-hole. The same procedure must be done to enable upstream encryption: Setting a local resolver as the upstream for Pi-hole, which then in turn is able to use DoH/DoT to an upstream server. This increases the susceptibility to errors and reduces the resiliency of the overall system by increasing its complexity. Also, the implementation and maintenance cost increase significantly. Unbound provides means of enabling DoT and DoH as via it's config files, at least when working with a current version of Unbound. DNSSEC is enabled by default in Unbound.

B. Problems And Common Risks

All three software setups suffer from the same problem regrading privacy: When the upstream must be secured with encryption, recursive name resolution cannot be done locally, it must be done by the upstream server and the local resolver must trust the answer. Fully encrypted recursive resolving would only be possible for a query, if all nameservers that would be recursively queried to answer the query support at least one of the types of transport encryption and the local resolver would opportunistically try to communicate over the secure channel with them. There is not standard procedure for detecting if a nameserver has such a secure channel, DoH even forbids it in the RFC [9]. Also, there is no standard way of secure recursive name resolving. This leaves the following risk: You must trust the upstream DNS provider – On data that is not DNSSEC signed, you have no way of telling if the data has been tampered with by the upstream server provider.

1) Confidentiality

DNS-over-HTTPS has one big advantage over DNS-over-TLS: It uses the same port as standard HTTPS, which makes it harder (but not impossible, i.e., you still must connect to a DNS resolver, by probing, it is still possible to identify it) to distinguish it from standard traffic. DoT in turn is, if using the standard port, detectable and blockable. Both protocols provide confidentiality, but DoH provides slightly better privacy.

Regarding Pi-hole: If chosen to not be complemented by additional software, as mentioned before, it does not provide confidentiality out-of-the box, neither on client-side, nor upstream. For some setups (i.e., for inexperienced users), this invalidates it as a deployment option.

2) Integrity

DNSSEC is supported by all three applications and only requires the upstream servers to support it, too. Unbound comes with already enabled DNSSEC validation, Pi-hole as well as AdGuard can be configured over the web interface to enforce it, too.

3) Availability

Even flooding the servers with large amount of DNS queries did not stop service operations (See Section C below). All servers have the same hardware specifications (2 Core CPU @ 2.3 GHz, 1 GB RAM). Resource exhaustion is very unlikely even at these levels when used in a local network. An attacker that can send packets to the local blocking resolver might try a DoS attack, the AdGuard software was observed to detect a host that sends large bursts of queries and slows its answering process – Pi-hole and Unbound do not do the same. Availability could be improved further by hosting two identical resolvers or even more to introduce redundancy and enable load distribution or load balancing.

C. Threat Evaluation

Validating incoming DNS answers from the upstream server via DNSSEC allows for reliable DNS cache poisoning protection [21]. For Domains that do not support DNSSEC,

the local server must rely on the answer from the upstream server and that it took measures to prevent DNS cache poisoning on their end. The communication channel between the local resolver and the upstream resolver can be protected by DoH or DoT.

Flooding attacks were conducted against all three applications by a single host with the following different attack types:

1) Sending out bursts of DNS queries to random domains that are known to not exist (NXDOMAIN Attack)

This was carried out against the default DNS endpoint UDP port 53 and DoH Port 443. Only AdGuard on Port 53 seems to have a mitigation technique in place: It considerably slows down the answering after the first few requests. The other two applications did not show this behaviour.

2) Sending out bursts of DNS queries to random subdomains that are known to not exists (Random Subdomain Attack)

This showed the same results as with the NXDOMAIN attack – only AdGuard with DNS over UDP slows the answering process down.

3) SYN TCP Flood to Port 53 and Port 443

The SYN segments were alle crafted the same (i.e., payload, sequence number). All applications performed similarly, only answering the first SYN and ignoring the following SYNs from the same host.

4) UDP Flood to Port 53

The same behaviour of the application like with SYN Floods shows on UDP Floods: Only the first datagram gets answered by all resolvers.

5) DNS Rebinding Check

AdGuard did not filter out DNS answers that contained private IPs but did set the TTL higher than it was received in the original answer (From 60 seconds to 300). Pi-hole did neither, it forwarded the original DNS answer with low TTL containing the RFC 1918 address. Unbound increased the TTL the same way as AdGuard and did also not remove the RFC 1918 address. Unbound allows for configuration of the minimum TTL via the configuration option *cache-min-ttl* and allows for direct mitigation of DNS rebinding via the *private-address* option.

V. Discussion and Conlusion

The standard DNS provides no security in the sense of confidentiality and integrity. It can not protect itself against attacks like MITM. There are extensions to the DNS protocol that aim to improve security:

DNSSEC provides integrity protection to the DNS protocol by introducing transaction level data and data origin authentication. It also provides a mechanism for denial of existence. DNSSEC allows to detect name-based authentication attacks but cannot protect against DoS attacks nor can it provide confidentiality. Also not further discussed here are the following points: Risk of compromise of DNSSEC keys, Zone enumeration and key rollover problems.

Confidentiality is provided by both DoT and DoH, which are ways to eliminate eavesdropping and tampering with data in transit. Combining one of these protocols with DNSSEC allows for confidentiality and integrity protected transfer of DNS data. This can be further combined, like described in this paper, by using these algorithms with blocking or filtering DNS resolvers:

Blocking DNS resolvers like Pi-hole and AdGuard provide means of blocking advertisements and malicious content to users by altering DNS answers or stopping DNS resolution for affected domains/names. Pi-hole cannot be directly configured to use DoH or DoT, neither upstream nor for clients. DNSSEC is directly supported. AdGuard supports DNSSEC, DoH and DoT directly. Comparing to the commonly used Unbound DNS resolver, which also supported these protocols directly, Pi-hole lacked functionality and therefore also security features. Integrating these features should be a primary task for the developers.

API support for implementation of a dynamic DNS blocking or filtering service was best supported by Unbound, where it is possible to directly check and alter DNS records via the API. AdGuard only allows for adding and removing names from the blocking list, after they already were requested at least once. Pi-hole does not provide an API, which renders it unusable for dynamic DNS filtering, it only supports static filter lists. API support should be expanded.

A. Integration and Deployment Into the Existing DNS Structure

Problematic is the integration into the existing DNS infrastructure. The local resolver might either have an upstream resolver that it trusts – like configured for this paper – or it might recursively lookup the requested domain name itself for the client. From a security standpoint, it might be wanted to let itself recursively ask the responsible nameservers for the correct resource record instead of trusting an upstream provider (which, for example, is configurable for Unbound with IP-network-based access-control). **Doing this in a confidentiality protected manner is currently not possible**. **Neither DoT nor DoH provide means of usage for direct requests to authoritative nameservers.** There is no mechanism for direct secure requests from a local recursive resolver to nameservers (with a fallback to insecure communication). DNSSEC can be used for recursive name resolution and this is configured to be enabled by default for Unbound.

DoT and DoH both can be used for upstream communication protection to an internet resolver (like Cloudflare, Quad9 or Google Public DNS) – but cannot be used for direct communication to nameservers – DoT provides an opportunistic scheme to test servers if the support it – DoH forbids this approach. There is no way for servers to indicate that they support a privacy enabled protocol like DoT or DoH. This also is a problem on client-side for the local resolver: Clients must be explicitly configured to use the secure communication channel; they cannot switch by themselves. For both problems a secure negotiation mechanism, that allows the client/resolver to detect, trust and automatically use a secure communication channel to respective upstream servers, is desirable. Browser manufacturers have begun to implement their own servers into browsers to secure at least this part of DNS communication by default [20]:

B. Regarding Confidentiality enabled Client Software

Modern browsers have started to make DNS requests via DoH independently of the operating system itself [20] [43]. This obviously enables the applications to circumvent the blocking and filtering of domains effectively. To solve this problem, there are different approaches:

Chromium for example will only use DoH, when the DNS provider, that is configured on the host, is contained in a curated list [44] of DNS providers, that also provide DoH and will only then upgrade to DoH from using the systems standard DNS feature. When using a local Pi-hole or AdGuard server, Chromium does not upgrade to DoH as the local server is not in the auto-upgrade list.

Mozilla Firefox uses a so-called *canary domain*: use-application-dns.net. If a local resolver like Pi-hole or AdGuard returns a negative DNS response (NXDOMAIN) without an error, Firefox detects the network as unsuitable for it's internal DoH protocol stack and will return to using the operating systems DNS servers. But Mozilla states: "The use of this domain is specified by Mozilla, as a limited-time measure until a method for signaling the presence of DNS-based content filtering is defined and adopted by an Internet standards body" [45]. Both Pi-hole and AdGuard answer with NXDOMAIN to signal that they implement additional features and cannot be replaced by the browsers own DNS provider.

C. Affecting Integrity With DNS Blocking

DNS blocking or filtering cannot be detected for non-integrity protected domain names but can be detected by a client if a domain name is integrity protected by DNSSEC and DNSSEC is correctly deployed to the domain. This can be done by the client if it requests DS records from the root to the requested domain name. This way the client can verify if a domain should be signed. If a blocking resolver blocks a DNS resource record, it also needs to block the corresponding RRSIG record. The client can detect if a domain should be signed, but the server answers with a wrongly non-signed record, or alternatively, answers with an NXDOMAIN, which in turn would have to be a NSEC or NSEC3 record in response (which by design would be signed). Even if the local resolver blocks DS records and all signing data for the parent zones, most clients have the DNS root key on their local system and know that the root zone is signed. There is a way for a local resolver to correctly sign the wrong answers generated by blocking: The clients need a DNSSEC key signing key they have in their trust store that is used by the local filtering resolver for signing altered answers. As DNSSEC is gradually rolled out further to the DNS, this might be needed in the future, if clients enforce DNSSEC validation.

D. Security and Risk Assessment

Most of the threats against DNS that are described in RFC3833 [21] can be prevented effectively by using DNSSEC combined with a transport security protocol like DoT or DoH. DNS blocking is nothing else than what is described in the RFC under Chapter 2.4 as "Betrayal By Trusted Server". Following this reasoning, DNS blocking might itself be seen as a security risk, especially when used for blocking names that would be perfectly valid and legal requests.

a) Common Attacks

Regarding common attacks like those described in before in Chapter II.B, combining at least the named mechanisms successfully prevents the following:

1. DNS Cache Poisoning is impossible for DNSSEC signed domain names when the local resolver enforces DNSSEC validation.
2. Man-in-the-Middle or Packet Interception Attacks cannot be performed on a confidentiality protected secure communication channel like provided by DoT or DoH (assuming there is a certificate pinning algorithm in use).

It does not protect against Denial-of-Service attacks like flooding attacks or Phantom Domain attacks.

b) Bigger Attack Surface/Points of Failures

The system also introduces new problems, and some Problems of standard DNS resolvers remain:

1. DNSSEC is complex to implement, and most domains do not support it yet [46]. For unprotected domains, attacks that target integrity, like Name Chaining are still possible.
2. Response packet size of DNSSEC signed responses is increased, this could be used in traffic amplification attacks.
3. DNSSEC validation increases the workload for the local resolver, this can be leveraged for Denial-of-Service.
4. DoT and DoH require each an additional open TCP socket for communication on the local resolver for clients. This increases the attack surface for common attacks like Denial-of-Service attacks, overflow attacks or bypass attacks.
5. Especially for Pi-hole, that does not support DoT or DoH natively, additional software must be used to add these features, which increases the attack surface even more.

VI. References

[1] P. Mockapetris, "DOMAIN NAMES - CONCEPTS AND FACILITIES," 11 1987. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1034. [Accessed 21 09 2021].

[2] P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION," 11 1987. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1035. [Accessed 21 09 2021].

[3] R. Elz and R. Bush, "Clarifications to the DNS Specification," 07 1997. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2181. [Accessed 21 09 2021].

[4] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "RFC4033 - DNS Security Introduction and Requirements," Network Working Group, 03 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4033. [Accessed 20 07 2021].

[5] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "Resource Records for the DNS Security Extensions," 03 2005. [Online]. [Accessed 21 09 2021].

[6] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "Protocol Modifications for the DNS Security Extensions," 03 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4035. [Accessed 21 09 2021].

[7] D. Conrad, "Indicating Resolver Support of DNSSEC," 12 2001. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3225. [Accessed 21 09 2021].

[8] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels and P. Hoffman, "RFC7858 - Specification for DNS over Transport Layer Security (TLS)," 05 2016. [Online]. Available:

https://datatracker.ietf.org/doc/html/rfc7858. [Accessed 20 07 2021].

[9] P. Hoffman and P. McManus, "RFC 8484 - DNS Queries over HTTPS (DoH)," 10 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8484. [Accessed 20 07 2021].

[10] PricewaterhouseCoopers, "IAB internet advertising revenue report - 2016 full year results," New York, 2016.

[11] Network Working Group, "RFC1034 - DOMAIN NAMES - CONCEPTS AND FACILITIES," 11 1987. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1034. [Accessed 19 07 2021].

[12] Google, "Google Public DNS," [Online]. Available: https://dns.google/. [Accessed 19 07 2021].

[13] Cloudflare, Inc., "1.1.1.1 -- The free app that makes your Internet safer.," [Online]. Available: https://cloudflare-dns.com/. [Accessed 19 07 2021].

[14] NLnet Labs, "NLnet Labs - Unbound - About," 2021. [Online]. Available: https://www.nlnetlabs.nl/projects/unbound/about/. [Accessed 19 07 2021].

[15] GitHub, Inc., "Stargazers · AdguardTeam/AdGuardHome · GitHub," 20 07 2021. [Online]. Available: https://github.com/AdguardTeam/AdGuardHome/stargazers. [Accessed 20 07 2021].

[16] GitHub, Inc., "Stargazers · pi-hole/pi-hole · GitHub," 20 07 2021. [Online]. Available: https://github.com/pi-hole/pi-hole/stargazers. [Accessed 20 07 2021].

[17] Cloudflare, Inc., "1.1.1.1 FOR FAMILIES," [Online]. Available: https://cloudflare-dns.com/family/. [Accessed 20 07 2021].

[18] AdGuard, "AdGuard Home: In-depth overview," 18 08 2020. [Online]. Available: https://adguard.com/en/blog/in-depth-review-adguard-home.html#dns. [Accessed 20 07 2021].

[19] Pi-hole LLC, "cloudflared (DoH) - Pi-hole documentation," 06 03 2021. [Online]. Available: https://docs.pi-hole.net/guides/dns/cloudflared/. [Accessed 20 07 2021].

[20] Mozilla Foundation, "Firefox extends privacy and security of Canadian internet users with by-default DNS-over-HTTPS rollout in Canada," 08 07 2021. [Online]. Available: https://blog.mozilla.org/en/mozilla/news/firefox-by-default-dns-over-https-rollout-in-canada/. [Accessed 20 07 2021].

[21] D. Atkins and R. Austein, "Threat Analysis of the Domain Name System (DNS)," 08 2004. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3833. [Accessed 22 07 2021].

[22] Cloudflare Inc., "DNS security | Cloudflare," [Online]. Available: https://www.cloudflare.com/learning/dns/dns-security. [Accessed 22 09 2021].

[23] Infblox Inc., "Top Five DNS Security Attack Risks and How to Avoid Them," 05 08 2013. [Online]. Available: https://www.infoblox.com/wp-content/uploads/2016/04/infoblox-whitepaper-top5-dns-security-attack-risks-how-to-avoid-them_0.pdf. [Accessed 22 09 2021].

[24] E. Borges, "The Most Popular Types of DNS Attacks," Securitytrails, 22 11 2018. [Online]. Available: https://securitytrails.com/blog/most-popular-types-dns-attacks. [Accessed 22 09 2021].

[25] C. Mitchell and S. Ariyapperuma, "3.1.2 Transaction ID Guessing," in *Security vulnerabilities in DNS and DNSSEC*, Egham, 2007, p. 2.

[26] J. Jeong, S. Park, L. Beloeil and S. Madanapalli, "RFC8106 - IPv6 Router Advertisement Options for DNS Configuration," 03 2017. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8106. [Accessed 21 07 2021].

[27] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "RFC4033 - DNS Security Introduction and Requirements - Section 3," Network Working Group, 03 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4033#section-3. [Accessed 20 07 2021].

[28] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, "RFC4033 - DNS Security Introduction and Requirements - Section 12," Network Working Group, 03 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4033#section-12. [Accessed 20 07 2021].

[29] NLnet Labs, "NLnet Labs Documentation - Unbound - unbound.conf.5," 09 02 2021. [Online]. Available: https://nlnetlabs.nl/documentation/unbound/unbound.conf/. [Accessed 22 07 2021].

[30] Internet Systems Consortium, "4. BIND 9 Configuration Reference — BIND 9 documentation | Content Filtering," 2021. [Online]. Available: https://bind9.readthedocs.io/en/latest/reference.html?highlight=rebinding#content-filtering. [Accessed 22 07 2021].

[31] Internet Systems Consortium, "4. BIND 9 Configuration Reference — BIND 9 documentation | Response Rate Limiting," 2021. [Online]. Available: https://bind9.readthedocs.io/en/latest/reference.html?highlight=rate%20limiting#response-rate-limiting. [Accessed 22 07 2021].

[32] Pi-hole LLC, "Pi-hole documentation | Pi-hole as All-Around DNS Solution," 19 01 2021. [Online]. Available: https://docs.pi-hole.net/guides/dns/unbound/. [Accessed 24 07 2021].

[33] M. Srivastava, "hosts(5) — Linux manual page," 2000. [Online]. Available: https://man7.org/linux/man-pages/man5/hosts.5.html. [Accessed 26 07 2021].

[34] Pi-hole LLC, "Telnet API - Pi-hole documentation," 05 02 2020. [Online]. Available: https://docs.pi-hole.net/ftldns/telnet-api/. [Accessed 26 07 2021].

[35] AdGuard Team, "AdGuardHome/openapi at master · AdguardTeam/AdGuardHome · GitHub," 20 07 2021. [Online]. Available: https://github.com/AdguardTeam/AdGuardHome/tree/master/openapi. [Accessed 26 07 2021].

[36] NLnet Labs, "NLnet Labs Documentation - Unbound - libunbound.3," 09 02 2021. [Online]. Available: https://www.nlnetlabs.nl/documentation/unbound/libunbound/. [Accessed 26 07 2021].

[37] NLnet Labs, "NLnet Labs Documentation - Unbound - Python libunbound docs," 2021. [Online]. Available: https://www.nlnetlabs.nl/documentation/unbound/pyunbound/. [Accessed 26 07 2021].

[38] Z. Vasicek and M. Vavrusa, "Local zone manipulation — pyUnbound v1.0.0 documentation," 12 01 2009. [Online]. Available: http://www.fit.vutbr.cz/~vasicek/nic-vip/pyunbound/examples/example6.html. [Accessed 26 07 2021].

[39] Internet Systems Consortium, "5. Advanced DNS Features — BIND 9 documentation," 2021. [Online]. Available: https://bind9.readthedocs.io/en/latest/advanced.html#dynamically-loadable-zones-dlz. [Accessed 26 07 2021].

[40] Stichting NLnet, "BIND DLZ Home," 2004. [Online]. Available: http://bind-dlz.sourceforge.net/. [Accessed 26 07 2021].

[41] T. Krizek and Others, "Overview - bind-dyndb-ldap - Pagure.io," 17 06 2021. [Online]. Available: https://pagure.io/bind-dyndb-ldap. [Accessed 26 07 2021].

[42] Internet Systems Consortium, "bind9/bin/tests/system/dyndb/driver at main · isc-projects/bind9 · GitHub," 22 05 2021. [Online]. Available: https://github.com/isc-projects/bind9/tree/main/bin/tests/system/dyndb/driver. [Accessed 26 07 2021].

[43] The Chromium Authors, "DNS over HTTPS (aka DoH)," [Online]. Available: https://www.chromium.org/developers/dns-over-https. [Accessed 22 09 2021].

[44] The Chromium Authors, "net/dns/public/doh_provider_entry.cc - Chromium Code Search," 2020. [Online]. Available: https://source.chromium.org/chromium/chromium/src/+/master:net/dns/public/doh_provider_entry.cc. [Accessed 22 09 2021].

[45] Mozilla Corporation, "Canary domain - use-application-dns.net | Firefox Help," [Online]. Available: https://support.mozilla.org/en-US/kb/canary-domain-use-application-dnsnet. [Accessed 22 09 2021].

[46] R. Lamb, "DNSSEC Deployment Report," 05 08 2021. [Online]. Available: http://rick.eng.br/dnssecstat/. [Accessed 05 08 2021].

VII. APPENDIX

A. Software

TABLE II. USED SOFTWARE VERSIONS AND REPOSITORIES

Software	Software Information		
	Version	***Repository***	***Remark***
Pi-hole	v5.3.1	official GitHub-Repository: https://github.com/pi-hole/pi-hole/releases	
Unbound	1.12.0-1	official GitHub-Repository: https://github.com/NLnetLabs/unbound/releases	
AdGuard Home	v0.106.3	official GitHub-Repository: https://github.com/AdguardTeam/AdGuardHome/releases	
knot-dnsutils	3.1.0	official Linux-Repository: deb http://ppa.launchpad.net/cz.nic-labs/knot-dns-latest/ubuntu focal main	Used for DoT and DoH requests to servers
dnspython	2.1.0 for Python 3.8	PyPI: https://pypi.org/project/dnspython/	Used for NXDOMAIN and sub-domain attack tests
Scapy	2.4.5 for Python 3.8	PyPI: https://pypi.org/project/scapy/	Used for flooding attack tests

Fig. 2. Listing of all software with the used version number (Older versions will most likely lack features)

B. Timings Data

Timing tests were conducted on a small scale on the three applications regarding answer timings on flooding attacks, NXDOMAIN and subdomain attacks. Results can be found here: https://www.kalytta.net/th-assets/results-timings.CSV. These are only for comparison and not clear enough to draw conclusions from it. Interestingly, AdGuard performed poorly on insecure requests. This was not further investigated.

C. Scripts

The scripts used for the tests can be found here: https://www.kalytta.net/th-assets/scripts/

D. Configuration Files

The configuration files for Unbound can be found here: https://www.kalytta.net/th-assets/configs/