2. DNS as a Component of an Organization's Security Strategy

DNS is critical to network connections, and its universal deployment makes it an effective security mechanism. The DNS platform is already in use by all types of clients on the network, including on-premises, in the cloud, and IoT devices. Thus, any protection provided by DNS infrastructure benefits all clients that use that infrastructure for name resolution, regardless of the type of device.

The original intent of DNS was to distribute information (e.g., host and IP address mappings, mail routing information), so it has not traditionally been viewed as a tool for securing network communications. However, because DNS enables nearly all network communications, it is an effective tool for monitoring and managing those communications. In a typical network communication pattern, the first action performed is to use DNS to resolve the domain name of the target system to an IP address. The communication is unable to begin until that resolution is completed.

According to the Cybersecurity & Infrastructure Security Agency (CISA), "DNS infrastructure is a common threat vector for attack campaigns" [30]. Even when DNS traffic is encrypted, the system still needs to retain access to the internet. This allows malicious actors to set up authoritative servers for command and control (C2) and data exfiltration, where encryption can work to their advantage. Therefore, it is crucial to implement strict controls and auditing on an organization's secure resolvers to ensure that only resolvers with the appropriate policy configuration are permitted to communicate with the internet. Applying security in DNS infrastructure gives administrators the opportunity to review potentially malicious communications before they begin and automatically prevent them from happening.

Two other advantages of using DNS are scale and efficiency. DNS has evolved over decades to support massive networks like the internet, so DNS security tools can handle a tremendous number of clients simultaneously. Name servers can load a large volume of authoritative and threat data. Taking protective actions with DNS is also efficient. Because DNS queries precede network communication streams, enforcing policy with DNS prevents malicious or suspicious communication streams from starting. Protective DNS decreases unauthorized traffic on a given network, which benefits the entire network infrastructure by alleviating the burden on other security elements, such as infrastructure components (e.g., firewalls) and human resources (e.g., the Security Operations Center [SOC]).

2.1. Protective DNS

Protective DNS[2] is enhanced with security capabilities to analyze DNS queries and responses and take action to mitigate threats. Protective DNS blocks access to malicious websites and prevents the delivery of malware, ransomware, phishing, and other attacks that attempt to deliver spyware and viruses. Protective DNS can be provided as a service from a vendor, deployed on internal DNS infrastructure, or a combination of the two. There are potential benefits to using a combination of externally provided Protective DNS with internally deployed

[2] This is the common used name for the described service and does not indicate the CISA program of the same name.

Protective DNS. While this approach may not be applicable in all cases, this combined hybrid scheme should be utilized where feasible.

The goals of deploying Protective DNS include:

- Blocking or redirecting harmful traffic in real time at the point of domain name resolution, typically before malicious activity starts
- Blocking categories of traffic with DNS by categorizing domain names that do not conform to an organization's policies or matching against known bad actor lists
- Delivering visibility into real-time and historical DNS query and response data to facilitate digital forensics and incident response
- Integrating with the wider security ecosystem as part of defense in depth, such as correlating an organization's data on assets (e.g., devices, cloud workloads) and users with the IP addresses of blocked queries
- Facilitating an organization's responsibility to comply with regulatory or contractual requirements for blocking traffic to disallowed sites (e.g., copyright violations, legal restrictions)

2.1.1. Threat Intelligence and Telemetry

As new networking technologies emerge and attack surfaces evolve, DNS remains a constant security control point for protecting users in all environments (e.g., organizations, mobile endpoints) and monitoring and disrupting malicious communications. Unlike other mechanisms in the security stack, it is not limited to any single type of threat and can often stop complex, multi-stage attacks before they progress. DNS can also protect users and organizations from scams, credential theft, ransomware, and data exfiltration.

This approach requires integrating threat intelligence into the DNS resolver. Threat intelligence is leveraged in a DNS infrastructure via mechanisms such as RPZs and can be seamlessly integrated into the DNS resolution chain via different architectures. Therefore, the consumption and deployment of threat intelligence services should be considered as part of any Protective DNS deployment.

2.1.2. Name Resolution Filtering

Name resolution filtering refers to DNS infrastructure that applies security-related policies to DNS resolution. For example, a Protective DNS implementation might refuse to resolve a set of domain names that are known to be used in phishing campaigns or that identify malware command-and-control infrastructure. Instead of resolving these domain names to IP addresses or other types of data, Protective DNS generally returns some other form of DNS response to indicate that the domain name does not exist, such as NXDOMAIN (i.e., "non-existent domain"). Protective DNS implementations can also log queries for domain names that trigger policy to indicate potential malware infection or other malicious activity.

Protective DNS is generally implemented by using RPZs configured on an on-premises DNS service; a cloud-based, secure recursive DNS service; or some combination of the two. Figure 1 shows an enterprise that utilizes a cloud-based service but retains a local forwarder for hosts on the enterprise local network.

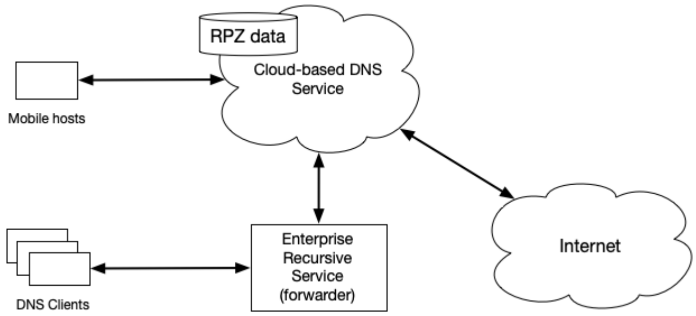

Fig. 1. Enterprise using cloud-based Protective DNS with a local forwarder

RPZs allow administrators to specify security policies that are enforced by the DNS service. For example, these policies could filter out specific domains or IP addresses. Many internet security organizations also provide “feeds” of their current threat data as RPZs. Organizations can apply those feeds to configure their name servers as secondary name servers for those RPZs and automatically synchronize their DNS resolution policy with the latest threat information. With an on-premises deployment of DNS, RPZs provide local control over these policies with very low latency, increased survivability, and minimal performance impact. With the ubiquity of DNS infrastructure and properties of RPZ, DNS is able to consistently mitigate threats across entire network infrastructures in a matter of minutes.

Secure recursive DNS services are also available on the internet. Organizations can configure their name servers to forward queries for internet domain names to the IP addresses operated by these services. The services generally offer a web-based control panel that allows administrators to customize the resolution policy applied to queries from the organization’s clients. The cloud approach offers greater scalability, storage, and computing power but has disadvantages, such as a loss of confidentiality, higher latency, and challenges in quickly and accurately attributing DNS queries to their sources. Organizations may consider using a combination of these to benefit from the lower latency of on-premises services and the greater scalability of cloud services while allowing for granular tracking, logging, and attribution of connections and requests. The best choice for a given DNS deployment will vary depending on the specific needs of the network and its users.

2.1.3. DNS for Digital Forensics and Incident Response

Government agencies and regulated enterprises should implement robust DNS traffic logging mechanisms to meet compliance requirements. Logging should capture both current and historical DNS traffic to enable digital forensics and incident response. These DNS logs should be integrated with other system logs to facilitate correlation with cloud workloads and device or user activities and to enhance visibility and auditability.

Logging all DNS traffic can be resource-intensive. If not done efficiently, it may impact the performance and availability of an organization's DNS services. If full DNS traffic logging is determined to be too resource-intensive, organizations may consider using cloud-based solutions, efficient logging methods (e.g., DNSTAP format [4]), or selective logging. However, DNS queries and the responses associated with domains that are classified as malicious or unauthorized by Protective DNS services should always be logged to support security and compliance objectives. An organization may remove known secure domains from logs to reduce volume and operating costs before sending them to a security information and event management (SIEM) system but should keep a complete log for future forensics.

Mapping an IP address to a compromised asset in the event of a cyber attack requires tracking key attributable metadata in real-time as well as a history of its allocation to each asset and resource, such as a Dynamic Host Configuration Protocol (DHCP) lease history. To ensure rapid notification of queries that might indicate infection or malicious activity, organizations should integrate Protective DNS logs from their name servers or their secure recursive DNS service with their SIEM or log analysis platform.

2.2. Protecting the DNS Protocol

DNS is a fundamental network service and must be left open to enable internet connections. As a result, it has been used by threat actors as a strategic vehicle to send malware and conduct data exfiltration, C2, and other attacks. The Internet Engineering Task Force (IETF) published an early threat model of DNS [5] that primarily focused on its query/response role. However, there are other considerations that require additional protective measures.

If a DNS server is compromised, there is little limit to the amount of short- or long-term damage that can be inflicted, often while avoiding detection. To that end, it is crucial to prevent bad actors from using DNS as a threat vector. There are two equally important elements to accomplishing this:

1. Protecting internal and external authoritative and recursive DNS services against threats
2. Usage Encrypted DNS and authentication to protect privacy and confidentiality

2.2.1. Protecting the Integrity of DNS Services

The DNS protocol refers to the standardized communications that carry DNS information between networked entities. Threats to the DNS protocol are numerous but well-studied. Section 3 discusses specific threats and protection approaches.

2.2.2. Using Encrypted DNS and Authentication to Protect the Protocol

There are tested mitigation methods available for securing the DNS protocol itself, including security focused on data (i.e., zone data) and channel security (i.e., DNS message transactions).

The DNS Security Extensions (DNSSEC) is a standardized set of extensions to DNS for securing DNS protocol communications against compromise. It is one part of the wider field of DNS security and must be implemented alongside other best practices and protocol features. While DNSSEC can help protect against the compromise of DNS communications, it does not provide any privacy protection (e.g., encryption). Those capabilities are provided by other technologies, such as DNS over Transport Layer Security (TLS) (DoT), DNS over Secure Hypertext Transfer Protocol (HTTPS) (DoH), and DNS over Quick UDP Internet Connections (QUIC) (DoQ) (see Sec. 4.2.1).

The process of authenticating the source of a message and its integrity through hash-based message authentication codes (HMAC) is specified through a set of DNS specifications known collectively as TSIG. The term “HMAC” denotes both the message authentication code generated by using a keyed hash function and the hash function itself. HMAC is specified in Request for Comment (RFC) 6151 [6] and generalized in the NIST document Federal Information Processing Standards Publications (FIPS PUBS) 198-1 [7]. The HMAC function for TSIG specified in RFC 8945 [8].

Later sections of this document categorize deployment guidance for securing the DNS protocol by type (i.e., authoritative in Section 3.8, recursive in Section 4.2.4, and stub resolvers in Section 5.1), including the use of DNSSEC and TSIG.

2.2.3. DNS Hygiene and Best Practices

Threat actors can exploit misconfiguration and lapsed domain/DNS resolver registration to seriously compromise DNS integrity. Organizations should implement robust processes to continuously monitor and validate the integrity of their public domains and take steps to raise the visibility of attempts to impersonate domains owned by the organization. Section 3 provides an in-depth breakdown of threats to DNS Hygiene and a discussion of best protection approaches.

2.3. Protecting the DNS Service and Infrastructure

DNS software must run on some existing host platform, which includes the hardware, firmware, and software that are required for DNS services to operate. Compromising any of these can potentially compromise the DNS service and cascade into significant operational failures or the loss of integrity and confidentiality. The following non-exhaustive list provides examples of compromises that could jeopardize the integrity of the host, platform, or software on which any given DNS deployment relies:

- **Platform or component vulnerability:** The OS, any system software, or any other application software on the DNS host could be vulnerable to attacks that result in the denial of name resolution services or threats to the hypervisor or underlying cloud infrastructure if the DNS service is cloud-based.
- **Distributed denial of service:** A malicious attacker could send a large volume of queries to perform a denial-of-service (DoS) attack against a server or service. The attacker

could also use numerous third-party DNS servers to aid in the attack (i.e., distributed DoS or DDoS).

- **Unauthorized configuration change:** The platform-level configuration file that enables DNS communication could be corrupted or subject to unauthorized modifications due to inadequate protections, resulting in disruptions varying from the breakdown of communication among DNS hosts to complete failure of the DNS service itself.

In this context, securing the platform and host refers to following the relevant best practices for securely deploying the non-DNS components on which the DNS relies (e.g., securely configuring the operating system that the service is running on, ensuring that the hardware supply chain is well-understood). Like any other network service, a DNS requires a large stack of diverse components to function, and providing the best practices for securing all of them is outside of the scope of this document[3]. However, the high-level architectural design of an organization's network infrastructure should be carefully laid out to protect high-criticality elements, including DNS services.

2.3.1. Dedicated DNS Services

Cyber criminals and other actors will seek to amplify and maximize the disruption of any cyber incident by attacking mission-critical systems, especially targets that host multiple critical components. To ensure cyber resiliency, the coexistence of multiple mission-critical services on a single system should be limited (i.e., separation of duties).

Even if a DNS is run on a secure operating system, vulnerabilities in other software programs on that OS can be used to compromise the security and availability of DNS software. Hence, the infrastructure that hosts DNS services should be dedicated to that task and hardened for this purpose to reduce the attack surface and ensure that adequate system resources are available to the DNS service. The infrastructure should include sufficient capacity for elements of the DNS service, such as logging, the support of encrypted DNS protocols, and Protective DNS. This may be easier to accomplish on purpose-built DNS services, either as a service or via virtual or physical appliances. If this is not feasible, then incorporating related core services (e.g., DHCP) may be appropriate.

Most name servers can be configured to perform authoritative and recursive functions. An authoritative name server serves resource records (RRs) from its own zone file or database. Since it is only meant to provide name resolution for the zones for which it has authoritative information, the security policy should have recursion turned off for this type of name server. In contrast, a recursive function would serve RRs from the name server's cache or by querying other name servers. It provides resolution services (i.e., resolving queries on behalf of clients), so protection can be ensured by restricting the number and location of clients from which it will accept queries.

---

[3] Refer to relevant documentation, standards, and best practices for any hardware, firmware, or software that is a dependency for a given DNS deployment.

A recursive name server should be run under a different security policy than that of an authoritative name server to mitigate attacks (e.g., cache poisoning). A name server instance that is directly accessible from or has direct access to the internet should be configured as either an authoritative name server or a recursive name server. Name servers that are only accessible from inside an organization's network commonly perform both functions — hosting internal DNS zone data and providing recursion for clients. In that context, combining the functions on the same server is acceptable and practical.

An organization could also classify its authoritative name servers into two sets based on the types of clients and data they serve (i.e., public versus private). External name servers would be located within a DMZ or any public-facing hosting service. These would be the only name servers that are accessible by external clients and would serve zones and RRs that pertain to hosts with internet-facing services. Internal name servers are usually located within the firewall perimeter and should be configured to be unreachable from outside of the organization's network (i.e., provide name resolution services exclusively to internal clients).

DNS software should not run or be present on hosts that are not designated as name servers. Some OS versions may come with DNS server software or other resolution components by default. Hence, while taking an inventory of organizational software in workstations and servers as part of the security audit, remove them from hosts that are not functioning as name servers.

2.3.2. Resiliency and High Availability of DNS Servers

As a critical service, DNS servers should be made as resilient and available as possible to ensure business continuity. This can be accomplished by using network and geographic dispersion and implementing best practices for server backups and recovery.

Every zone must have an authoritative primary server and one or more authoritative secondary name servers. At least two of the authoritative name servers for an organization should be located on different network segments. This dispersion ensures the availability of an authoritative name server when a particular router or switch fails or during an attack on an entire network segment (e.g., DoS/DDoS attack). Authoritative name servers should also be dispersed geographically (i.e., different physical sites). For example, organizations may locate some authoritative name servers on their own premises and others in a hosting service or partnering organizations. Other solutions could rely on cloud-hosted DNS services, which handle the availability of zone information for the organization.

If the organization hosts the zone information, a network administrator should use a "hidden" primary authoritative server and only have secondary servers visible on the network. A hidden primary authoritative server is an authoritative DNS server that does not appear in the name server (NS) resource record set (RRset) for a zone. All of the name servers that do appear in the RRset as designated name servers must have a copy of the zone data, whether by zone transfer or some other method (e.g., database replication). In effect, all visible name servers are secondary servers, which prevents potential attackers from targeting the primary name server. A hidden primary should only accept zone transfer requests from a specified set of secondaries and refuse all other requests for zone transfers and, ideally, other DNS queries.

2.3.3. Interoperability of the Protective DNS Ecosystem

When an organization deploys a Protective DNS service, it must ensure interoperability with the wider security ecosystem so that it is additive to the overall security model. This includes:

- Ensuring that the DNS service is part of a “strength in depth” approach and not an isolated control
- Logging query/response data and blocks/redirects to security operations functions (e.g., via a SIEM/SOAR) to be correlated with other security events
- Being able to access the threat intelligence deployed within the Protective DNS service via APIs for use in assessments, forensics, and incident response
- Ensuring that DNS components use standardized and interoperable protocols and APIs as defined by the relevant IETF RFCs
- Sharing and forwarding relevant DNS data to other components of the wider security ecosystem

3. Managing Threats to Authoritative Services

The primary role of an authoritative DNS server is to provide answers to queries for the zones for which it is authoritative. These servers contain files known as zone files, which in turn contain information, such as name-to-address mappings (i.e., A RRs for IPv4 IP addresses or AAAA RRs for IPv6 addresses) or address-to-name mappings (i.e., PTR records).

There are two types of authoritative name servers: primary name servers and secondary name servers. A primary name server contains zone files that are created and edited by the zone administrator. Sometimes, a primary name server is configured to allow the zone file to be dynamically updated by authorized DNS clients. A secondary name server also contains authoritative information for a zone, but its zone file is a replication of the one in the associated primary name server. The replication is most often enabled through a standardized transaction called a zone transfer, which transfers RRs from the zone file of a primary (or other secondary) name server to the secondary name server. The secondary name servers are notified of any changes to the contents of a zone file on the primary name server through a transaction called *DNS NOTIFY*. When a secondary name server receives this message, it initiates a zone transfer request to the primary name server (or the configured secondary from which it is zone transferring). Depending on the circumstances, the zone transfer can contain the entire zone file (i.e., AXFR) [9] or the incremental changes since the last AXFR (i.e., IXFR) [10]. To improve fault tolerance, there are usually several secondary name servers in an organization.

When the authoritative role is provided by a cloud-based service, there is often no differentiation between primary and secondary roles. Cloud-based DNS authoritative hosting services may use means other than zone transfers to synchronize DNS data across cloud instances. From a client perspective, there is no difference between a primary or secondary role providing answers to a query since they both provide authoritative answers for the zone. The primary role is only relevant to clients when they are attempting to update DNS records via DNS UPDATE (sometimes called *nsupdate)* [11], which must be sent to the primary for the zone.

Domains that are hosted on authoritative DNS infrastructure must be protected against exploitation. Misconfigured or lapsed registrations of Canonical Name (CNAME) RRs or name server delegations allow threat actors to take control of an organization’s external-facing domains. This allows the threat actor to use the positive reputation of those domains in attacks against the organization’s own users (i.e., spear phishing) or as part of general malware campaigns. Threat actors increasingly register “look-alike” domains that are not owned by the target organization but which users could easily assume to be associated with that organization.

3.1. Zone Transfer Threats and Protection Approaches

Zone transfers replicate zone files to multiple servers to provide a degree of fault tolerance in the DNS service provided by an organization. While threats from zone transfers have not been formally documented through any IETF RFCs, a common threat to any network transaction is the denial of service. A second common threat to any network packet is based on the exploitation of knowledge gained from the information provided by zone transfers.

- Denial of service (DoS): Because zone transfers involve the transfer of entire zones, they place substantial demands on network resources relative to normal DNS queries. Errant or malicious frequent zone transfer requests on name servers can overload a primary or secondary zone server and result in the denial of service to legitimate users.
- Unauthorized zone modification: The zone transfer response message could be tampered with.

A DoS threat can be minimized if secondaries allowed to make zone transfer requests are restricted to a set of known entities. Configuring this restriction into the primary and secondary name servers requires a means of identifying those entities. IP addresses are commonly used but are not secure because they can be spoofed.

The IETF developed an alternate mechanism called a transaction signature (TSIG) [8], whereby the mutual identification of servers is based on a shared secret key. Because the number of servers involved in zone transfer is generally restricted to name servers in the same administrative domain of an organization, a bilateral trust model that is based on a shared secret key may be adequate for most organizations. TSIG specifies that the shared secret key be used for both mutual authentication and for signing zone transfer requests and responses to protect against tampering.

Asymmetric cryptography (i.e., public-key cryptography) can also be used to authenticate DNS transactions. The format of the SIG(0) RR [12] is similar to the resource record signature (RRSIG) RR and can be validated using a public key stored in the DNS instead of a shared secret key. While SIG(0) can be more computationally expensive to use, a previous trust relationship may not be necessary to use SIG(0) signed messages. However, because most zone transfers occur between parties that have a previously established relationship, it is considered easier to implement TSIG for authenticating zone transfer transactions.

A lower-level network layer solution (e.g., IPSec or other secure network communication technologies) can also provide security and remove the need for authentication at the application layer. The protocol for DNS zone transfer over TLS ensures that zone transfers are encrypted [13].

3.1.1. Restricting Zone Transfer Transaction Entities

Authoritative name servers (especially primary name servers) should be configured with an access control list that designates the hosts from which zone transfer requests will be accepted. These restrictions address DoS threats and potential exploits from the unrestricted dissemination of information about internal resources. Hence, zone transfer from primary name servers should be restricted to secondary name servers. The zone transfer should be completely disabled in secondary name servers unless they are also intended to provide zone transfers to other secondary name servers. In that case, the zone transfers on the secondary should also be protected with access control lists (ACLs).

3.2. Zone Content Threats and Protection Approaches

DNS data is made up of zone information and configuration information. All of the security deployment options discussed in this section relate to zone file contents, particularly:

- Parameter values for certain key fields in RRs of various RR Types
- The presence of certain RRs in the zone file

The various types of data in the zone file result in different security exposures and potential threats.

3.2.1. Lame Delegations

Incorrect zone delegation (i.e., lame delegation) (see Appendix B) can result in a child zone becoming unreachable (i.e., denial of service) or intermittently accessible if only one or more NS RRset entries points to a non-existing server.

3.2.2. Zone Drift and Zone Thrash

There may be a mismatch of data between the primary and secondary name servers if the Refresh and Retry fields in the start of authority (SOA) RR [2] of the zone are set too high and the zone file is changed frequently. This error is called *zone drift* and results in incorrect zone data at the secondary name servers. If the Refresh and Retry fields in the SOA RR are set too low, the secondary server will initiate zone transfers frequently. This error is called *zone thrash* and results in more workload on both the primary and secondary name servers. Such incorrect data or increased workload may result in degradation or the denial of service.

The Refresh value should be determined by the expected rate of change for a zone. Suggested values (in seconds) usually range from 1200 (20 minutes) to 432000 (12 hours) or even 864000 (1 day). The Retry value is the time that a secondary should wait before a failed refresh and should be a fraction of the Refresh value for the zone.

3.3. Dynamic Update Threats and Protection Approaches

3.3.1. Dynamic Update Misuse

Dynamic updates involve DNS clients making changes to zone data in an authoritative name server in real time [11]. As with zone transfer transactions, the threats associated with dynamic update transactions have not been officially documented by the IETF in an RFC. However, the following common threats could be expected:

- **Unauthorized updates** could have several harmful consequences for the content of zone data, such as:
    - Adding illegitimate resources
    - Deleting legitimate resources

    - Altering delegation information (NS RRsets pointing to child zones)
- Update tampering could affect data in a dynamic update request
- Replay attacks (i.e., update request messages captured and resubmitted later) could cause inappropriate updates

3.3.2. Guidance on Securing Dynamic Updates

Unauthorized or tampered updates could be countered by authenticating the entities involved and providing a means to detect message tampering. The TSIG mechanism meets these security objectives for zone transfer and is specified for protecting dynamic updates. Although the dynamic update message contains some replay attack protection in the prerequisite field of the message, TSIG provides an additional protection mechanism by including a timestamp field in the dynamic update request. This signed timestamp enables a server to determine whether the timing of the dynamic update request is within the acceptable time limits specified in the configuration. Security can also be enhanced using a lower-level network layer mechanism, such as IPSec.

Dynamic updates on a zone file can only be directed to the primary name server, which holds the writable copy of the zone file. Dynamic update requests generally originate from hosts (e.g., DHCP servers) that attempt to provision a new host name in the DNS zone when an IP address is dynamically assigned to a client. Because dynamic update messages change the authoritative zone data, they should only be accepted from authorized senders (e.g., TSIG, ACL).

3.4. DNS NOTIFY Threats and Protection Approaches

3.4.1. DNS NOTIFY Misuse Threats

DNS NOTIFY is a message sent by primary name servers that cause secondary servers to start a refresh operation (i.e., query for SOA RR to check the serial number) and perform a zone transfer if an update to the zone has occurred. Because the NOTIFY message is only a signal, there are only minor security risks in dealing with the message. The primary security risk to consider is spurious NOTIFY messages.

3.4.2. DNS NOTIFY Protection

Receiving spurious DNS NOTIFY messages results in an increase workload for secondary name servers because a zone transfer will only occur when an updated zone is on the primary server. Because this threat is low-impact, the required protection approach is to configure the secondary name servers to receive DNS NOTIFY messages only from the zone’s primary name server. However, if TSIG is set up for use for all communication between a set of hosts, TSIG will be used with NOTIFY messages as well.

Once zone transfers have been set up between servers, secondary name servers should be informed about changes to zone file data through a notification message. By default, a DNS

NOTIFY message is sent to every recognized secondary name server of the zone (i.e., name servers listed in the NS RRset in the zone) whenever a primary name server detects a change in the zone file. Most DNS server software provides the ability to also notify other servers, which allows the administrator to account for any stealth name servers for the zone (i.e., not listed in the NS RRset). DNS administrators should keep notifications on because this configuration will allow updates to be propagated quickly to secondary name servers.

3.5. Minimizing Information Leakage

As part of operational security, it is important to minimize the amount of information that can be gathered off of a DNS server without credentials. This information can be used to launch attacks on the DNS server or to quickly learn a larger organization's network.

Not all information leakage is preventable. Some information stored in DNS servers must be public in order for a DNS to function correctly. This section provides discussions and recommendations on how to best reduce information leakage without compromising core functionality.

3.5.1. Resource Record Information

Attackers can map an organization by using RRtypes to learn about available services, such as mail exchange (MX), server selection (SRV), TLSA, and text strings (TXT). Types of RRs in the DNS that are meant to convey information to humans and applications about the network, hosts, or services include the responsible person (RP) record, the host Information (HINFO) record, the location (LOC) record, and the catch-all TXT RRtype [2]. Although these record types are meant to provide information to users in good faith, they also allow attackers to gain knowledge about network hosts before attempting to exploit them. For example, an attacker may query for HINFO records to find hosts that list an OS or platform known to have exploits. Therefore, a best practice is to exclude these record types from internet-facing zones.

More careful consideration should be given to the TXT resource record type. A DNS administrator will have to decide whether the data contained in a TXT RR constitutes an information leak or is a necessary piece of information. For example, several authenticated email technologies (e.g., sender policy framework [SPF], domain keying for internet mail [DKIM], domain-based message authentication, reporting and conformance [DMARC]) use TXT RRs to store email sender policy information, such as valid email senders for a domain [14]. These judgments will have to be made on a case-by-case basis.

3.6. External Authoritative Domain Integrity

Threat actors often target legitimate public-facing DNS domains to reduce suspicion and improve the efficacy of their phishing and malware campaigns.

3.6.1. Dangling CNAME Exploitation

When a DNS CNAME record links two domain names together, there is the risk that the parent domain of the canonical name that the record points to does not remain registered by the target organization. As a result, threat actors can register the parent domain and cause DNS resolutions to resolve to the threat actor's controlled domain. CNAME records can also be exploited if the canonical name resolves to an IP address that is no longer in use by the domain owner, and the attacker can gain control of that IP address to conduct attacks.

DNS administrators should develop policies and procedures to regularly monitor and assess the configurations and registrations of these domains. When they are no longer required, CNAME records should be deleted.

3.6.2. Lame Delegation Exploitation

A lame delegation can result in domain hijacking. When a subdomain is delegated to a DNS-hosting provider and the contract for providing DNS services for that domain lapses, threat actors could hijack resolution for that subdomain by contracting with the provider that controls the servers targeted by the delegation to host that subdomain under their control. This then enables the threat actor to redirect resolution requests to their own infrastructure.

DNS administrators should actively validate that there are no lame delegations within their external authoritative domain name space and use DNS-hosting providers who apply safeguards.

3.6.3. Look-Alike Domain Exploitation

Threat actors extensively leverage look-alike or *typosquat* domains to impersonate target organizations. By leveraging the positive reputation of legitimate organizations, threat actors vastly increase the success rate of their phishing and malware campaigns. These look-alike domains can include subtle variations of legitimate domains or text or character substitution to register a domain that appears to be owned by a legitimate organization.

A common best practice is to monitor new DNS registrations to detect this attack vector and to defensively register look-alike domains if feasible. Gaining visibility into these activities will enable organizations to preemptively address a prevalent threat vector that targets their users and consumers.

3.7. Operational Recommendations

3.7.1. Resource Record TTL Value Recommendations

Each RRset in a zone has its own time-to-live (TTL) value that tells recursive DNS servers and clients how long (in seconds) the RRset data should be stored in its cache upon receipt, although not all clients cache responses or strictly obey TTLs. When a recursive server receives a DNS response to a query and performs all relevant checks, it stores the resulting RRsets in its cache. The server cache decrements the TTL value of each RRset in its cache, and that data is purged from the cache (excluding DNS services with prefetching or similar mechanisms) when the TTL for any RRset reaches zero. This prevents caches from growing too large, removes old and possibly incorrect DNS data from caches, and prevents stale results from being returned to future queries.

The zone administrator can assign the default TTL value for all RRs in the zone, but each RRset can have its TTL individually specified, and different RRsets in the same zone can have different values. The zone administrator should set the default TTL value long enough to ensure that the RRset will be useful for caches but short enough that any changes to the RRset will be propagated quickly through the DNS (and old information purged). DNSSEC signature validity periods should also be taken into consideration. TTL values should be less than the validity period of the RRSIG that covers the RRset. DNSSEC-aware clients will decrement the TTL value of an RRset in its cache to the signature expiration date if that date is before the projected TTL. That way, the RRset will be purged before the signature expires and seen as BOGUS by other DNSSEC validators. However, DNSSEC-unaware clients may not know to do this comparison, so there is a risk that invalid DNSSEC RRsets will be stored in DNSSEC-unaware caches.

TTL values should be in the order of hours with a recommended range of 1800 (i.e., 30 minutes) to 86400 (i.e., 1 day). If a zone administrator knows that the DNS data is likely to change frequently, the TTL value could be set lower for those specific records to ensure that old, stale data is purged from server caches. If the zone administrator believes that the DNS data will not change frequently, then the TTL value can be set higher to optimize the benefits of caching in recursive servers. While some specialized load-balancing scenarios rely on much shorter time periods (i.e., 60 seconds or less), 30 minutes to 24 hours is sufficient for most DNS data. If the data is signed using DNSSEC, the value should always be long enough to ensure that the data will not be purged from caches before validating resolvers can validate it. Very low TTL values (i.e., 30 seconds or less) can cause problems with DNSSEC-validating caches and should be avoided for DNSSEC-signed RRsets. Even for non-signed zones, extremely low TTL values should be avoided. In particular, a TTL of 0 should never be used because it has been known to cause a multitude of issues. Even a TTL in the range of 5-30 is significantly better than 0.

3.8. **DNSSEC Signing Considerations for Authoritative Service**

DNSSEC refers to a set of protocol extensions that add source authentication and integrity protection to DNS data [15]. In DNSSEC, a digital signature covers a given RRset in a response and is encapsulated through a special RRtype called RRSIG. The keys used to validate these digital signatures are also stored in the DNS in DNSKEY RRsets. Trust in the public key is established by building a chain of validated signatures and public keys that are sometimes operationally differentiated as zone-signing keys (ZSK) and key-signing keys (KSK) from signed DNS data to a trusted public key that is preconfigured on the system. The preconfigured public key of the trusted zone is called the *trust anchor*.

DNSSEC can guarantee the integrity of name resolution response data to DNS clients that perform DNSSEC signature verification. DNSSEC does not protect the confidentiality of DNS query/response data. Confidentiality can be protected using DoH, DoT, or DoQ (see Section 4.2.1).

Deploying DNSSEC for an authoritative DNS service requires a set of steps to digitally sign the zone data and configure the authoritative service [16]. The exact process for these steps depends on the implementation used by the organization.

Due to the naming convention, DNSSEC is often conflated with the wider and more general concept of DNS security. However, DNSSEC is only one component of a larger whole, and more tools must be utilized to achieve more comprehensive DNS security.

3.8.1. DNSSEC Key Considerations

DNSSEC is based on public-key cryptography to generate digital signatures over DNS data. The current set of cryptographic algorithms is defined in RFC 8624 [17], which lists the mandatory and optional algorithms supported by DNSSEC tools. Using RFC 8624 and the guidance on key strength and lifetimes in SP 800-57 [18] and SP 800-131A [19] produces the recommended digital signatures algorithms shown in Table 1.

Table 1. DNSSEC key parameters based on algorithm

Signing Algorithm	DNSSEC code	Public Key Length (in bits)	Signature length (in bits, no RRSIG encoding)
RSA with SHA-256	8	2048 bits or greater	~2050-4100
ECDSA P-256 with SHA-256	13	256	512
ECDSA P-384 with SHA-384	14	384	768
Ed225516	15	256	512
Ed448	16	456	912

SP 800-57 gives required security strengths and lifetimes for cryptographic key material based on intended use. A strength of 112 bits is acceptable until 2030, when it will be raised to 128 bits. This is for DNSSEC signing and not validating, as using keys (or algorithm suites) for backward compatibility with entities that do not have the same cryptographic requirements is allowed. Additionally, some organizations (e.g., federal agencies) have additional requirements around the use of FIPS 140-certified cryptographic modules [20].

As there are restrictions to the total size of a DNS response (without resorting to using TCP), administrators should be aware of how the key and digital signature size affect the DNS response size. Therefore, algorithms like ECDSA and ed448 are preferable over RSA, as they produce smaller key and signature sizes. Table 1 does not include post-quantum cryptographic (PQC) algorithms. At the time of writing, the use of PQC for DNSSEC has not been specified. However, administrators should consider migrating to PQC algorithm usage when the use of PQC algorithms has been specified, tools have been updated to include the new algorithms, and the majority of DNS clients support them. ECC algorithms with smaller key sizes would be more vulnerable to a quantum attack, as it would require a currently theoretical quantum computer with fewer qubits than would be required for an RSA key with the same cryptographic strength [21]. Since DNSSEC does not provide confidentiality, the risk of "store and future decrypt" with quantum computers would not be a threat.

The key lifetime value is the period during which a key pair should be considered active and used. Afterward, the key is retired and no longer used to generate signatures. Signing keys used for DNSSEC are categorized as signature keys with a recommended maximum lifetime of 1-3 years. The act of retiring a signing key pair and introducing a new signing key pair is called a *key rollover* in DNSSEC. As with other DNSSEC operations, there are automated tools available to perform key rollovers with minimal administrator intervention based on the enterprise policy on key lifetimes. Key lifetime values do not have a representation in DNSKEY or RRSIG RRsets. They are completely internal to the enterprise and set by local policy.

3.8.2. Using RRSIG Validity Periods to Minimize Key Compromise

The best way for a zone administrator to minimize the impact of a key compromise is by limiting the validity period of RRSIGs in the zone and in the parent zone. This strategy limits the time during which an attacker can take advantage of a compromised key to forge responses. An attacker that has compromised a ZSK can only use that key during the KSK's signature validity interval, and an attacker that has compromised a KSK can only use that key during the signature interval of the RRSIG covering the DS RR in the delegating parent. Therefore, ZSK validity periods should be kept short to require frequent resigning. KSK changes and changes with the registrar should be infrequent (e.g., once per year). To reduce the chance that the KSK is compromised, strong modern algorithms (e.g., ECDSA) should be used.

3.8.3. Hashed Authenticated Denial of Existence

A side effect of the DNSSEC security extensions as they were first specified is the ability for a user to "walk" a zone by sending a series of queries for NSEC RRs. A client could send a query for an NSEC RR in the zone and "walk" the zone by sending a follow-up query for the NSEC RR to the next name indicated in the received NSEC RR. This would result in a client being able to enumerate the entire contents of a zone. While this is not an attack by itself (all DNS data is considered public), it would most likely be a prelude to an attack. An attacker could enumerate a zone to discover the IP addresses of servers to attack directly. This concern led to the creation of NSEC3, but the emergence of attacks like KeyTrap (CVE-2023-50387[4]) has brought into question whether the computational effort that NSEC3 requires is worthwhile. It appears that since the emergence of KeyTrap, the more prudent approach is to use NSEC. If the use of NSEC3 is still required due to local policy, the NSEC3 parameters should be set to minimize the DoS risk [22].

[4] CVE-2023-50387 detail from NIST National Vulnerability Database: https://nvd.nist.gov/vuln/detail/cve-2023-50387 (accessed Jan 2025)

3.8.4. DNSSEC Algorithm Migration

It may eventually be necessary to migrate to a new DNSSEC signing algorithm due to a discovered weakness in the currently used algorithm or overriding policy decisions. Migrating from the current DNSSEC algorithm to a new algorithm requires a set of steps and delays while old data is removed from caches.

A proposed process can be found in RFC 6781 [16], which outlines the basic steps. To reduce the risk of a validator thinking it is under a downgrade attack, the signatures for the new algorithm are added before the DNSKEY RR with the new algorithm public key. Likewise, when removing the retiring algorithm, the public key DNSKEY RR is removed first, followed by the signatures.

DNSSEC-enabled responses can grow large enough to trigger fallback to TCP during the algorithm rollover period and while two RRSIG RRs are present for every RRset in the zone. Administrators need to consider their response size and the complexity of operations when initiating an algorithm rollover.

3.8.5. DNSSEC Signing Internal Zones

Using DNSSEC to sign internal zones is generally considered bad practice. In the few instances where internal zones must be signed, it is likely that the administrators already know that it is required and why. Any existing regulatory or contractual requirements that include the signing of internal zones should be reviewed.

There are two distinct functions that DNSSEC can perform: signing and validating. Signing allows a domain owner to provide its customers with assurance that they are getting the correct DNS data from the true owner. Validating is performed by recursive resolvers to check the authenticity and integrity of the DNS responses it receives. The signing of zones should be confined to public-facing external namespaces only, and validation should be enabled on any recursive servers.

There are three major reasons why signing internal zones is discouraged:

1. Depending on the namespace, it is generally difficult if not impossible to tie the internal zone back to the internet chain of trust. Because the chain will be broken, validating the internal zones will require the management of additional trust anchors for each zone on all of the validating resolvers.

2. If the zone is being dynamically updated, the entire zone will have to be re-signed each time it is updated, which can be computationally intensive. For large or busy zones, frequent re-signing could lead to the degradation or complete denial of DNS services for clients.

3. DNSSEC-signing the internal zones serves no purpose when authoritative and recursive services are combined on the same DNS servers, which is a common and acceptable deployment for internal DNS systems. In this type of architecture, the DNS server would be doing nothing other than validating itself.

Therefore, the general guidance would be that internal-only zones should not be DNSSEC-signed. Any exceptions to this guidance are beyond the scope of this document.

4. Recursive/Forwarding Service and Stub Resolvers

A recursive name server for an organization performs the name resolution function on behalf of a collection of clients. A recursive name server may also be called a caching name server or a recursive resolver [23]. The name resolution function is performed by a recursive name server in response to queries from a stub resolver. The search process for name resolution may involve searching its own cache, forwarding to other designated name servers, recursively querying various authoritative name servers through a set of iterative queries, or a combination of these methods (see Appendix A).

Some name server implementations can be configured to be both an authoritative and a recursive name server. In this configuration, the same name server provides authoritative responses for queries that pertain to authoritative zones while it performs the resolving functions for queries that pertain to other zones. To perform the resolving function, the server must be configured to allow recursive queries, which makes it more vulnerable to attack than a server that does not allow such queries. Since authoritative information might be compromised, it is not a good security practice to configure an internet-accessible name server (often referred to as an external name server) to perform both authoritative and recursive functions. However, it can be an acceptable and efficient configuration for purely internal name servers that are not internet-accessible.

A recursive service is accessed via an IP address, which could be a local recursive resolver or a cloud-based service (public or private). Cloud-based services can have the advantage of global availability and larger, more active caches since they serve millions of clients. However, there may be trade-offs in control and visibility when an organization decides to rely solely on public recursive services.

4.1. Threats to Recursive/Forwarding Service

Compromised transactional queries and incoming responses can threaten recursive servers. A forged or bogus response is different from those expected from legitimate authoritative name servers. Threats to the recursive service could include:

- A compromised authoritative name server (for queries that originate from a recursive name server)
- A poisoned cache of a recursive name server (for queries that originate from a stub resolver or a forwarding recursive name server)
- A recursive name server that is induced to query for a specific name, and forged responses are immediately sent to the server attempting to get a malicious answer in the cache (mitigations have historically been put in place to prevent this)
- Specific queries that expose bugs in name server software implementations to launch a DoS attack or impact operations in some way
- A passive monitor that can observe DNS queries sent from a stub resolver, which could lead to a loss of privacy for end users of the host system or a monitor learning what

applications or services are running on the host in order to identify communication patterns between the host and network services

- Improper DNSSEC management by the domain owner (e.g., during key rollovers), although this is not due to malicious activity

The cache of a recursive (caching) name server could be poisoned by the following attacks:

- **Packet interception.** The attacker eavesdrops on a request and sends a response by spoofing an authoritative name server before the real response from the legitimate authoritative name server reaches the recursive name server.
- **ID guessing and query prediction.** The attacker guesses the ID field in the header of the DNS request message (e.g., using brute force guessing) and possibly the query name (QNAME) and query type (QTYPE). The attacker then injects bogus data into the network as a response by spoofing a name server.
- **Responses from a compromised authoritative name server.** A compromised authoritative name server is directed by a controlling adversary to send out bogus responses to queries from recursive name servers.
- **Incorrect expansion rules applied to wildcard RRs.** Many zones use wildcard RRs to economize on the volume of data in the zone file. The wildcard patterns are used for synthesizing RRs to generate responses to queries as described in RFC 1034 [1] and RFC 4592 [24]. If synthesis rules are applied incorrectly in a name server, the RRs associated with organizational resources existing may not be generated or made available in a DNS response. This fault also results in a denial of service.
- **Removal of RRs from a response.** An attacker could also remove RRs from a response, which may result in a name resolution failure and consequent denial of service.

4.2. Recommendations for Protection

DNSSEC was designed to provide authentication for DNS data and does not protect the confidentiality of DNS query/response transactions. As originally specified, DNSSEC and non-DNSSEC query/response are sent unencrypted and are vulnerable to passive monitoring. This allows a third party to observe the queries being sent by a stub resolver on a host system, see what network services the host is communicating with, and develop a possible “fingerprint” of the device (or end user) for later tracking. Some organizations’ security services also use DNS monitoring to detect unauthorized communication or potential malware operating within an organization.

4.2.1. Encrypted DNS

Communication between stub resolvers and the recursive DNS servers they query has traditionally been unencrypted. The DNS messages exchanged by stub resolvers and DNS servers have a binary encoding that is widely understood and easily decoded. This communication has, therefore, been subject to both interception and spoofing, which can reveal sensitive information or allow an attacker to redirect unsuspecting users to malicious sites.

To address these threats, the IETF has developed several enhancements to DNS that are collectively known as Encrypted DNS. Each protocol enhancement can encrypt communications between stub resolvers and recursive DNS servers in slightly different ways:

- DNS over Transport Layer Security (TLS) (DoT) [25] runs the traditional DNS protocol over TLS, which is the same layer of encryption used to secure traffic between web browsers and web servers that use HTTPS to communicate. TLS, in turn, runs over TCP.
- DNS over HTTPS (DoH) [26] runs the DNS protocol over HTTP, which in turn runs over TLS. TLS runs over TCP.
- DNS over Quick UDP Internet Connections (QUIC) (DoQ) [27] runs DNS over QUIC, which is an encrypted transport layer that runs over UDP.

All of these protocols optionally allow recursive DNS servers to authenticate themselves to stub resolvers, addressing the threats of interception and spoofing.

4.2.1.1. Encrypted DNS Guidance and Recommendations

The U.S. Government requires the DNS infrastructure of Federal Civilian Executive Branch (FCEB) agencies to support the use of encrypted DNS when communicating with agency endpoints, wherever technically supported [31].

Many organizations will find that their options to implement Encrypted DNS protocols are limited by the protocols that their applications or operating systems currently support. Organizations may also need to configure applications (e.g., browser software) that use encrypted DNS so that local resolvers that act as a point of control and logging are not bypassed. Some endpoints (e.g., IoT devices) may not have the necessary software modules to use encrypted DNS on their own and may require a forwarding DNS server that acts as a recursive service, as shown in Fig. 2.

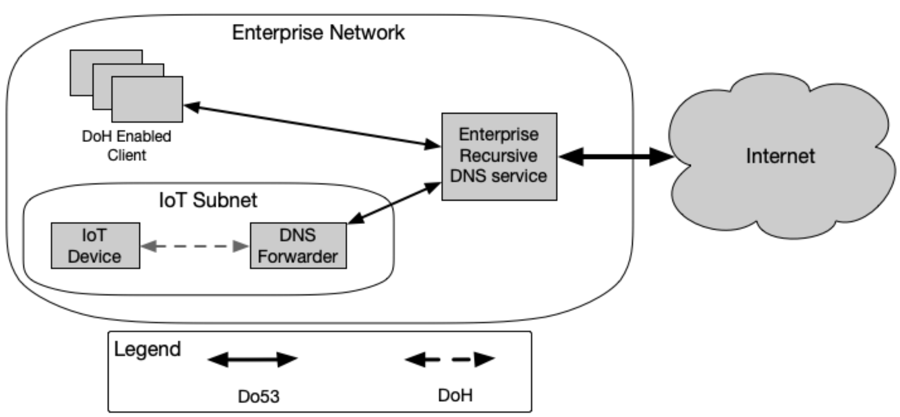

Fig. 2. Mixed use of DoH and Legacy DNS over UDP port 53 (Do53)

While this is not a complete zero trust architecture, it still allows for strict policies on outgoing DNS traffic. Also, depending on architecture needs of responsiveness and availability of the DNS service, the Enterprise Recursive Resolver could be implemented as a cloud service.

4.2.1.2. Considerations for Using Encrypted DNS

Encrypted DNS is crucial for enhancing online privacy and security. Encryption helps protect sensitive information from being exposed or manipulated and reduces the risk of attacks (e.g., DNS spoofing, man-in-the-middle attacks). It is a vital component in broader organizational strategies for securing internet communications.

However, Encrypted DNS introduces additional overhead, particularly on name servers, because of the need to perform encryption and decryption when sending and receiving DNS messages, respectively. Organizations should anticipate this and ensure that their name servers have sufficient resources to handle their query load before beginning any widespread deployment of Encrypted DNS.

The use of Encrypted DNS may also make troubleshooting more difficult because IT staff using network troubleshooting tools will not have ready access to the contents of DNS queries or responses, though that information will still be on the name servers themselves.

4.2.1.3. Cryptographic Guidance

All varieties of Encrypted DNS support server authentication, but this requires the configuration of a server certificate on each name server that receives queries over Encrypted DNS. This certificate can be generated and supplied by either an internet or internal certificate authority.

If supported, name servers must be configured to only allow the cryptographic ciphers permitted in SP 800-52 [28], and cloud-based Encrypted DNS providers should be assessed for this. Additionally, some organizations may have requirements on the use of FIPS-140 approved cryptomodules for use with TLS or DTLS [20], which would also include Encrypted DNS.

4.2.2. Restricting the Use of DNS With Public Providers

In order to ensure that users do not use unauthorized public, internet-based DNS services, organizations should:

- Block outbound DNS from the internal network to the internet, except for name servers that are authorized to communicate directly with name servers on the internet (e.g., forwarders). This blocking can be implemented using firewall rules or router access control lists (ACLs) because DNS uses two well-known ports: UDP port 53 and TCP port 53.
- Restrict stub resolvers to only use authorized DNS services to leverage encrypted DNS wherever possible.

- Block unauthorized DoT traffic from the internal network to the internet using firewall rules or router ACLs. DoT is straightforward to block because it uses the well-known TCP port 853.
- Block unauthorized DoH traffic from the internal network to the internet using RPZs and firewall rules. DoH is more difficult to block because it uses the same TCP port as TLS: 443. RPZs can help block the resolution of the domain names used to identify known DoH servers, and firewall rules can block access to public DoH services that run on dedicated IP addresses.
- Use mobile device management (MDM) or other central management solutions to prevent users from configuring non-approved external encrypted DNS services.

4.2.3. Detecting and Mitigating Data Exfiltration via DNS

Organizations should establish controls to detect and block unauthorized applications from tunneling data within DNS packets. Signature-based systems can enable the detection of well-known DNS tunneling tools, but customized DNS data exfiltration tools should also be considered as threat actors increasingly turn to this tactic to avoid detection by signature-based systems.

4.2.4. Enabling DNSSEC Validation

Configuring DNSSEC validation requires two tasks: configuring the server to perform validation and configuring one or more trusted public keys to act as trust anchors. The method for achieving this depends on the DNS implementation used.

The policy for determining which DNSSEC public keys to configure as trust anchors is beyond the scope of this guide and would be part of the organization’s security policy. However, given the hierarchy used in DNS, the higher the public key in DNSSEC, the wider range of DNS responses can be validated using that key, as shown in Table 2.

Table 2. Trust anchor selection

Example Public Key	Could Validate
root “.”	All signed TLDs and below
.gov	All signed names under “.gov”
Example.gov	Only names in the “example.gov” domain

Therefore, the internet root key should be configured as a trust anchor.

4.2.5. Maintaining DNSSEC Trust Anchors

When a zone updates its DNSSEC signing keys (i.e., performs a key rollover), any validating recursive resolver that has configured that zone’s key as a trust anchor must obtain the new singing key. If not, DNSSEC-signed responses from the zone (or delegated zones) will fail DNSSEC validation.

There is an automated process for a zone to signal to validating recursive resolvers that it is performing a key rollover [29]. Administrators may not get advanced notification of a rollover, which makes relying on manual trust anchor updates risky. Validating recursive resolver administrators should enable automated trust anchor rollover and monitor logs to ensure stability in the recursive service.

4.3. Operational Recommendations

There are additional steps that an administrator can take when setting up a recursive server for an organization. Because it is unwise to allow queries from the internet to the recursive server, the recursive server can be placed behind a firewall that blocks inbound connections from UDP and TCP port 53 (used by DNS).

Recursive servers should be configured to only accept queries from internal hosts and perform recursion for them. Different recursive DNS server implementations may have features to enable this ability. This could be done by implementation specific ACLs or network infrastructure configuration, which is beyond the scope of this document.

5. Stub Resolvers

Software that require access to the internet or internal network resources (e.g., web browsers, email clients, cloud applications, operating systems) use a DNS client called the client resolver, resolver library, or stub resolver. A stub resolver is often referred to as simply a client. The stub resolver formulates a name resolution query for the resource sought by the network-accessing software and sends it to a recursive name server in the enterprise (see Appendix A). Stub resolvers are generally configured to send queries to two or more recursive name servers to provide some fault tolerance for their operation.

The OS-layer stub resolver is a centralized DNS client within the operating system that handles DNS queries for all applications by sending them to a recursive resolver configured at the system level (e.g., via network settings). In contrast, an application-layer stub resolver operates independently within a specific application and bypasses the OS resolver to handle DNS queries directly. This is common in modern browsers or applications that support DoH or DoT for enhanced privacy and security. These two resolvers often interact when an application-layer stub overrides the system's default resolver settings, which can create conflicts or complementarities, depending on how they are configured. For instance, an application might use its own DNS settings for specific queries while still relying on the OS stub for others, creating a layered approach to DNS resolution.

5.1. Securing the Stub Resolver

Stub resolvers do not have many configuration options. Often, the only configuration necessary for an administrator is to enter the IP addresses of one or more recursive resolvers that the stub would rely on to resolve queries. It is a good idea for administrators to include the IP addresses of at least two recursive servers to increase the availability of the DNS service for end users. This can be done manually or using a protocol like DHCP. Additional resiliency can be provided to stub resolvers through the use of anycast for the DNS service. By adding a layer of abstraction between the IP of the DNS service and the individual servers via anycast, an administrator can lower the risk of client impact from the loss of any one DNS server. This can be especially important for supporting certain application stacks that will only use one DNS server, regardless of the stub resolver configuration.

There is a known class of malware that attempts to change the settings on a system's stub resolver to direct queries to another (usually malicious) recursive server. The server may then direct end users to a malicious site where another attack takes place, or the server may simply direct users to a web page that serves ads or similar non-intended content. To combat this, administrators should make sure end user systems have the latest endpoint protection software and consider blocking all outbound DNS traffic with the exception of known recursive servers.

Implementing enterprise mobility management (EMM) software can ensure that stub resolvers are correctly configured to point to authorized DNS servers. Additionally, EMM facilitates the management and enforcement of policies regarding the approval and use of software with integrated stub resolvers within the enterprise environment.

5.2. DNSSEC Considerations for Stub Resolvers

Most stub resolvers cannot perform DNSSEC validation and may not understand DNSSEC at all. These stub resolvers will have to rely on an upstream validating recursive resolver to perform DNSSEC validation on its behalf. This does not pose a significant risk on a trusted organization's network because there are several options to protect the link between a validating recursive resolver and a stub resolver that cannot do DNSSEC validation processing.

If an organization's network is considered trusted (e.g., using one of the last hop mechanisms or similar), then the stub resolvers can be considered to be using DNSSEC. However, network administrators should be aware that DNSSEC validation failures could complicate the diagnosis of internet error messages. DNSSEC validation failures will be seen by the upstream validator, not the stub resolver that initiated the query. The stub resolver may only see a generic server failure message (SERVFAIL), which applications interpret differently. Network administrators should check validator logs when responding to network errors to rule out DNSSEC validation failures.

5.2.1. Recommendations for Providing Service to Mobile Hosts

Mobile or nomadic hosts present a particular challenge for network administrators. These systems often access a network outside of the trusted enterprise, so mobile hosts must either perform their own validation or have a trusted connection to an enterprise-approved DNS service. If the mobile hosts can perform their own validation, then the same policy for the enterprise validators should be applied for the mobile host. That is, the same trust anchors and validation policy should be set for mobile hosts as for validators on the enterprise network.

It may be necessary for a mobile system to configure its validator when migrating from the enterprise to an external network and vice versa. If the enterprise network is trusted, the mobile host can rely on the enterprise validator when on the enterprise network and perform its own validation when on external networks. Ideally, network administrators can avoid this problem by using alternative names for internal and external zones, thus having different trust anchors.

If the mobile host cannot perform its own validation, it must either have a secure tunnel back to the enterprise network or a secure connection to an approved DNS recursive service. Many enterprises already have a means for mobile hosts to access internal resources (e.g., file servers), so a validating recursive server should be added as one of the services provided to mobile hosts through a secure channel. Alternatively, enterprise mobile hosts could be configured to use an approved cloud-based DNS recursive service to allow mobile hosts to use approved DNS recursive services without needing to have a connection to the enterprise infrastructure.