DNS Security Risks and Best Practices

by Bilal Ibrahim, Senior Network Engineer, UncommonX

The domain name system (DNS) has been an essential element of modern networking for years. It translates domain names to internet protocol (IP) addresses so people can access sites through their browsers. Since DNS operations will carry private information, DNS risk has several layers involving the management of the internal and external DNS requests.

Those risks vary depending on the working environment of the many co-working operating systems running on networking devices, servers, workstations, and other end user devices. In general, private network environments commonly point to a pair or more of internal DNS servers that handle private-to-private DNS requests and private-to-public DNS requests.

Standard DNS queries are required for almost all web traffic. They create various opportunities for security DNS exploits such as on-path attacks and DNS hijacking where a website's inbound traffic can be redirected to a fake copy of the site. This type of attack can expose businesses to major liabilities.

To help you reduce risks and secure your system, here are common attacks associated with DNS and the best practices for preventing them.

Types of DNS Attacks

DNS hijacking: This redirects queries to a different DNS server. It's done either with malware or by the unauthorized modification of a DNS server. The result is like DNS spoofing. Fundamentally, this is different attack than spoofing because it targets the DNS record of the website on the nameserver instead of the DNS server's cache.

DNS spoofing/cache poisoning: This attack introduces forged DNS data into a DNS resolver's cache, causing the resolver to return an incorrect IP address for a domain. Instead of going to the correct website, traffic can be diverted to a malicious machine for the purpose of distributing malware or collecting login information.

DNS tunneling: Attackers can use ports for SSH, TCP, or HTTP to pass malware into DNS queries to tunnel or pass-through DNS queries and responses that are undetected by most firewalls.

Phantom domain attack: This is like an NXDOMAIN attack on a DNS resolver. The attacker introduces several "phantom" domain servers that either slow responses to requests or stops them completely. The resolver is then hit with a flood of requests to these domains, which ties it up, leading to slow performance and a denial-of-service attack.

Botnet-based CPE attack: This type of attack is carried out by using customer premises equipment (CPE), the hardware given out by service providers for use by their customers, including modems, routers, cable boxes, etc. If attackers compromise the CPEs then the devices become part of a botnet. The botnet in turn is used to perform random subdomain attacks against sites or domains.

Random subdomain attack: Attackers send DNS queries for several nonexistent, random subdomains for a legitimate site with a goal to create a denial-of-service for the domain's authoritative nameserver. This makes it very difficult to look up the website from the nameserver. The ISP serving the attacker may also be impacted in that a recursive event can be triggered, causing the resolver's cache to be loaded with bad requests.

Domain lockup attack: Attackers formulate special domains and resolvers and then create TCP connections with other legitimate resolvers. When the targeted resolvers send requests, these domains send back slow streams of random packets that slow the resolver's resources.

NXDOMAIN attack: An attacker floods a DNS server with requests to cause a denial-of-service of legitimate traffic. This type of attack can be launched using advanced attack tools like autogenerate unique subdomains for each request. NXDOMAIN attacks can also target a recursive resolver. The purpose is to fill the resolver's cache with junk requests.

DNS Security Best Practices

DNS infrastructure: Make sure that only the information necessary for the parties using the server is available on a DNS server. This includes allowing internal access only to DNS servers and all other DNS data and restricting access to the public. Publicly accessible servers should be authoritative-only without acting recursively. This will limit access to any individuals outside your organization.

Ensure availability: The DNS server should be in partly high-available. Set them up in a (HA) pair or cluster. If one fails, others will be able to assume the load. Whether it is a primary or secondary nameserver, recursive, or authoritative, a high-availability cluster can ensure availability for any DNS server resource. In the case of publicly accessible servers, provide geographically diverse servers in their domain name registration to prevent physically localized events. Routing diversity, with a preference from providers with unique autonomous system numbers (ASNs), can protect against large denial-of-service attacks that can affect a provider's ability to serve DNS data.

Hide primary servers: The servers that host the master copy of any zone should be hidden primaries. They should only exist to serve that data to the secondary nameservers throughout the organization. The primary server should not be listed as nameservers for any zone or be accessible to end users. Secondary nameservers are then meant to answer queries from primary nameservers. This helps to ensure the integrity of the DNS data. Limit access to the primary nameservers to just those individuals responsible for the maintenance of the servers and the data that resides on them. External nameservers should be behind a firewall with rules in place that allow the primaries nameservers to make queries and transfers.

Use nameservers as local for users: For example, a company with multiple branches or regional offices should set up both recursive and authoritative nameservers on-site to those nameservers serving those locations. This distributes the query load across multiple servers. This also helps to ensure that names are resolved as quickly as possible.

Restrict access of zone transfers: This will be protected by on-server access control lists (ACLs) along with transaction signatures (TSIGs). Limit connections to the nameservers from only the hosts used by those employees responsible for their maintenance and upkeep and not just through privileged account management. Nameservers serving authoritative data that are not also serving as recursive servers help to ensure availability by limiting their attack surface. Secondaries should be configured to deny all zone transfer requests. If a nameserver is serving authoritative data, then that server should not be used to serve as recursive server. All traffic to the nameserver should be restricted via ACLs on the server itself. Firewall-based ACLs can also be used to limit the traffic to the server, thus preventing certain classes of attacks like denial-of-service attacks, while ensuring that traffic that does reach the server is authorized to use the service.

Protect data integrity: Domain name system security extensions (DNSSEC) should be deployed to ensure the integrity of the data being served, particularly with publicly accessible zone data. DNSSEC digitally is used to sign DNS data so that nameservers can ensure the integrity of data prior to providing answers to queries. Fully deploying DNSSEC helps ensure end users are connecting to the actual website and other services connected to that domain name. This happens as described in accordance with the Internet Corporation for Assigned Names and Numbers (ICANN). The verification takes place through public key infrastructure (PKI). Here, digital certificates from the root server to the nameserver form a chain of trust between the very top of the DNS tree and the lowest end nodes (i.e., the end user's nameserver).

About the author

Bilal Ibrahim is an IT professional with more than 16 years of experience in the vast area of networking technologies. His background has largely been in financial services with hands-on experience in enterprise networks specializing in Cisco LAN/WAN/backbone/data center and security environments. Bilal leads the internal network design and automation for UncommonX.