Cybersecurity Threat Alert
NXNSAttack
June 1, 2020

Summary

On May 19, 2020, academics from the Tel Aviv University and The Interdisciplinary Center in Israel discovered a vulnerability in the implementation of DNS recursive resolvers that can be abused to launch disruptive DDoS attacks against any victim. The attack leveraging the vulnerability has been dubbed NXNSAttack by the researchers and detailed in their research paper.[1]

Unlike DDoS floods or application-level DDoS attacks that directly target and impact a host or a service, the NXNSAttack targets the domain name resolution capability of its victims. Like the NXDOMAIN or DNS Water Torture attack,[2] the DDoS attack is aimed at disrupting the authoritative servers of the domain by overloading them with invalid requests using random domain request floods through recursive DNS resolvers. This attack is hard to detect and mitigate at the authoritative server because the requests originate from legitimate recursive DNS servers. By disrupting name resolution for the domain, attackers effectively block access to all services provided under the domain. New clients will not be able to resolve the hostname of the service while under attack because they have no way of locating the IP address to connect to the service.

Unlike the limited 3x packet amplification factor of the NXDOMAIN attack, the NXNSAttack provides packet amplification factors ranging from 74x when attacking a subdomain (victim.com) up to 1621x when targeting a recursive resolver. The bandwidth amplification factors range between 21x for subdomain attacks and 163x when targeting a recursive resolver. Targeting root and top-level domain servers results in a packet amplification factor of 1071x and a bandwidth amplification factor of 99x. With high amplification rates and flexible targeting, NXNSAttack is a very capable attack vector which can be performed at scale.

Researchers have since disclosed the vulnerability and approached vendors and providers who have already patched their software and servers. The following DNS server implementations had a fix available at the moment of disclosure: ISC BIND (CVE-2020-8616), NLnet labs Unbound (CVE-2020-12662), PowerDNS (CVE-2020-10995), and CZ.NIC Knot Resolver (CVE-2020-12667). In addition, the following open DNS recursive resolver providers have updated their services to mitigate the use of the vulnerability for DDoS attacks: Cloudflare, Google, Amazon, Microsoft, Oracle (DYN),Verisign, IBM Quad9, and ICANN. Other software and service providers have followed the announcement with fixes and patching. However, it is safe to assume that not all recursive resolvers, private and public, have been or ever will be patched.

The exposure to attacks or abuse of the vulnerability is not limited to just public recursive resolvers but also impacts private recursive resolvers located at ISPs, clouds or within organizations. Malicious actors have leveraged different kinds of bots in the past to launch random domain flood attacks and can leverage the same bots to conduct a NXNSAttack which disrupts any victim outside of the resolvers' owners. Easy access to source code for botnets such as Mirai that provide "out-of-the-box" support for random domain floods adds to the potential to perform these disruptive DDoS attacks.

The victims have no immediate grasp on the risk they are exposed to. Any component of the authoritative DNS infrastructure, including the second level domain (victim.com), top level domain (.com, .info, …), and root name servers ('.') can be disrupted through recursive DNS resolvers that are outside of their control. Victims are at the mercy of DNS service providers.

Recursive DNS providers can protect their own infrastructure and protect the internet from attacks by applying the fix provided by DNS software suppliers or by implementing the Max1Fetch solution proposed by the researchers in their paper.[1] Alternatively, recursive DNS providers can protect their infrastructure against random domain floods through the aggressive use of DNSSEC-validated cache (RFC8198) or by leveraging Radware DefensePro for DNS protection.[3]

[1] The NXNSAttack paper is available from the researchers' website: http://nxnsattack.com/
[2] 2017's 5 Most Dangerous DDoS Attacks & Steps to Mitigate Them
[3] Protecting DNS Critical Infrastructure Solution Overview

DNS over HTTPS (DOH) or DNS over TLS (DOT) does **not** providing protection against the NXNSAttack. The DOH and DOT protocols are aimed at providing privacy on the client side of the name resolving and does nothing to protect the authoritative side of the DNS infrastructure. Worst case, DOH and DOT can be leveraged as evasion techniques to hide random domain name floods from upstream network sensors and protections inside TLS encrypted data streams, rendering detection or mitigation of malicious DNS attacks impossible.

Increasing the Time To Live (TTL) value of the domain zone will increase the resistance of services under the domain against authoritative domain server disruption, but will do so at the cost of the agility of the domain. Moreover, this will only provide a solution for those clients behind resolvers that have cached the resolution at an earlier time and will not provide a complete or indefinite solution whilst the attack is ongoing.

Resourceful and pervasive attackers can create an attack infrastructure to target any subdomain (victim.com), potentially impacting other subdomains provided by the same domain name server or service providers. Given enough resources, attacks can target top level domains such as '.com', '.info', '.us', '.ca', '.de', etc. and even attempt to disrupt the internet's root name servers.

Background

The attacks on Dyn's DNS infrastructure on October 21st, 2016 was a reminder of how individuals and businesses have come to rely on the availability of the domain name system. A well orchestrated botnet attack was able to disrupt a large portion of the internet leveraging a known DNS attack vector: the Water Torture attack. Thousands of IoT devices infected and orchestrated by the Mirai botnet impacted several high-profile internet services via a simple flood of random domain name requests through their local ISP DNS resolvers.

To translate hostnames such as www.radware.com to an IP address where a client should connect to consume the service, the internet uses a hierarchical domain name system consisting of authoritative servers which act as a large distributed phone book while recursive DNS resolvers closer to the edge of the internet provide the gateway to the phone book. Recursive DNS resolvers act as the middleman between clients and the authoritative DNS servers and take requests from clients to resolve hostnames in IP addresses. Recursive DNS resolvers cache hostname resolutions locally to provide faster responses to often requested hostnames, preventing repeatedly stressing the authoritative infrastructure. The large distributed cache is what keeps the authoritative DNS infrastructure alive.

The authoritative DNS infrastructure is built in a tree-like fashion (Figure 1) with a number of fixed and known root servers (".") containing the location information of Top Level Domain (TLD) name servers such as '.com', '.net', '.ca', '.info', etc. The TLD name servers contain references to all Second Level Domain name servers which are the authoritative subdomain name servers such as 'radware.com', 'microsoft.com', 'bitbucket.org', etc.

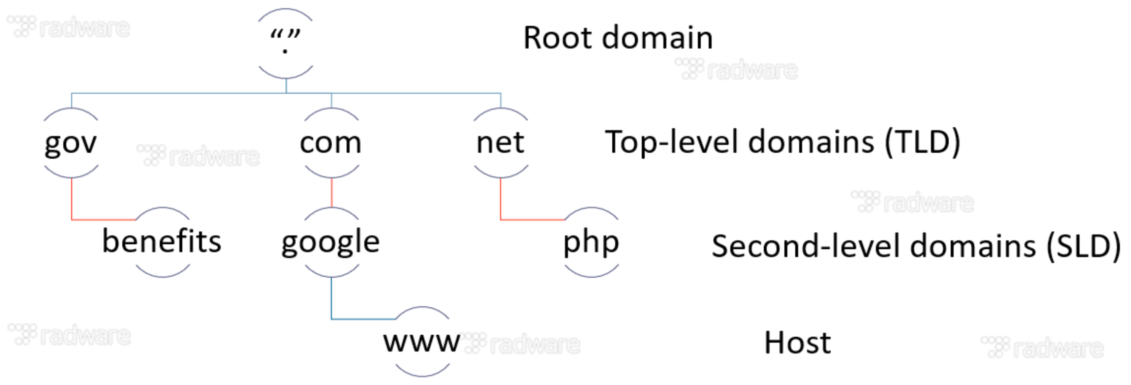


Figure 1: Hierarchical domain name infrastructure

To resolve a hostname (Figure 2), the client's operating system or internet router was configured with the IP addresses of recursive name servers that can be located within your ISP, alternatively they can be public open resolvers such as 1.1.1.1 (Cloudflare) and 8.8.8.8 (Google). The recursive resolver will start at the top of hierarchy, the root servers, to ask who would be able to provide the IP address of for example 'www.microsoft.com'. The root name servers are fixed and configured with their IP addresses in each recursive resolver and they will respond to the request by delegating the resolution to one or more top-level domain name servers. The delegation happens through a DNS referral response message which contains the names of TLD name servers capable of resolving the hostname. In the case of our example 'www.microsoft.com', the root name server will provide hostnames of several '.com' TLD name servers (see Figure 2, step 4). The recursive resolver continues its quest and contacts one of the received TLD server names to request the location of 'www.microsoft.com' (see Figure 2, step 5). The TLD will in his turn respond with another referral message (see Figure 2, step 6), delegating the request to the authoritative domain name server for the '.microsoft.com' domain, which in this case could be 'ns1.msft.net'. The recursive resolver then finalizes his quest by asking the 'ns1.msft.net' authoritative name server to provide a resolution for 'www.microsoft.com' (see Figure 2, 7) which responds with an 'A' record containing the IP address of the hostname (see Figure 2, step 8). The recursive resolver then caches the result to provide faster answers to subsequent requests (see Figure 2, step 9) and forwards the 'A' record on to the client (see Figure 2, step 10) which can then contact the host based on its IP address and start interacting with the service.

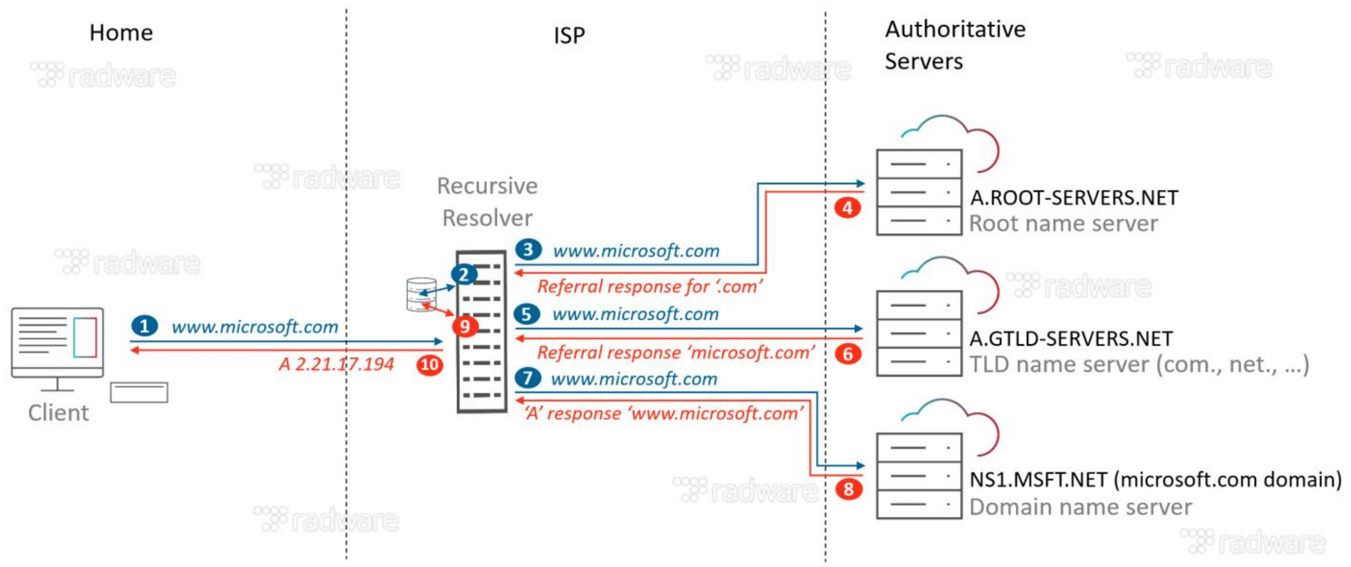
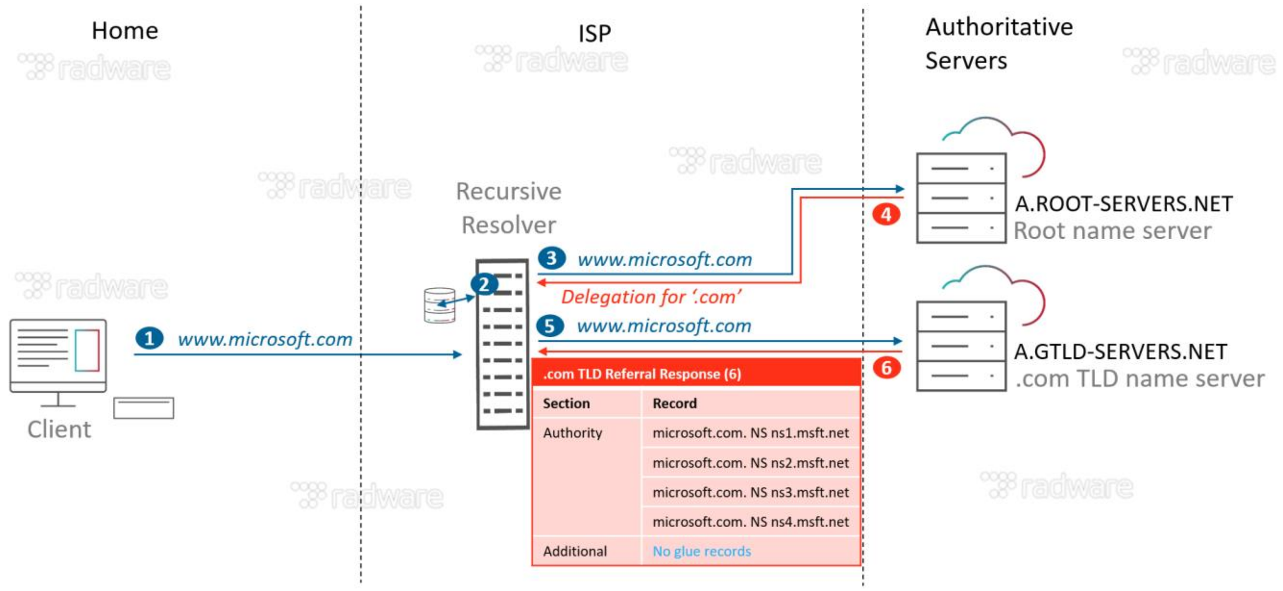


Figure 2: Hostname resolution process

See Figure 3 for an example of a referral response. The recursive hostname resolution process starts from the assumption that the recursive resolver knows the IP address of the hostnames that were delegated to by the root and TLD name servers in step 4 and 6. In reality, however, before the recursive resolver can continue its quest for 'www.microsoft.com', it first must start a new recursive request to resolve each TLD name server in the referral response received from the root server. The root server always responds with more than one server in the authority section of the referral response. The recursive resolver will have to go through each and every name server listed to resolve the name and then, based on round-trip time metrics of the different servers, decide which one is the fasted TLD name server and pass his original query on to that name server. The TLD name server then follows the same process as the root server and creates a referral response with the second level domain servers that can continue the resolution process of 'www.microsoft.com'.

Figure 3 contains an example of the referral response sent by the '.com' TLD for the 'www.microsoft.com' request. You will note that the referral response delegates the resolution to name servers located in the 'msft.net' domain. This is very much a legitimate response and in this particular case, the recursive resolver needs to go back to the root server and restart the process through the '.net' TLD name server to find the location of the name servers ns1.msft.net through ns4.msft.net. Remember that along the way, any repeated request to hostnames that were resolved in the recent past will be honored by the local cache, speeding up the process and reducing the number of requests needed to recurse the whole tree over and over again.



.com TLD Referral Response (6)

Section	Record
Authority	microsoft.com. NS ns1.msft.net
	microsoft.com. NS ns2.msft.net
	microsoft.com. NS ns3.msft.net
	microsoft.com. NS ns4.msft.net
Additional	No glue records

Figure 3: Example referral response from TLD name server for Microsoft.com

Readers will have noticed the 'Additional' section in the referral response in Figure 3 which contains the mention 'no glue records'. Glue records are a way for the authoritative name server to expedite the name resolution process on the resolver side by passing the 'A' record for all or some of the delegated hostnames in the authority section. The recursive resolver must, however, take care in which 'A' records it trusts. Imagine a malicious authoritative server for the domain 'attacker.com' passing a glue 'A' record with the IP of the server www.bankofamerica.com and the resolver just trusts it and caches that record for future resolutions. That is what is referred to as DNS cache poisoning. To prevent this, a rule was devised known as the Bailiwick rule. All recursive resolvers honor at least some form of the **Bailiwick rule** which states that a glue record should only be accepted by the resolver when the sender of the record is authoritative for the hostname in the record or from a different perspective, if the hostname in the record is within or below the zone or domain of the referrer.

The Bailiwick rule, while crucial for security, has a serious impact on the resolution process by the recursive resolver. Since not all SLD name servers are within the same zone or record, but are what we call "out-of-Bailiwick," the recursive resolver still needs to resort to manual recursion through the tree to resolve the delegated hostnames in the referral response. How many recursions will depend on the hostname and how the domain has been organized. To give some ballpark numbers: to resolve a host in the Microsoft.com domain a resolver exchanges 54 packets with the authoritative DNS infrastructure while a host in the twitter.com domain will require 388 packets.

The NXNSAttack

In the previous section we noted that when a recursive resolver receives a referral response from an authoritative server, it will go through each delegated hostname to resolve that name through a recursive process to an IP address. While doing so, the recursive resolver can record the round-trip time (RTT) and will use this metric to choose the best (fastest) name server to continue the resolution of the original query. Now imagine an authoritative server owned by a malicious actor, forcing the recursive resolver to go through a long list of delegated name servers by crafting a malicious referral response. In doing so, the malicious actor can maximize the load and number of exchanges between the recursive resolver and the authoritative servers for a single client query. Those are the foundations of the NXNSAttack and by crafting referral responses with different parts of the delegated hostnames randomized, the attack can be targeting the recursive resolver, the authoritative victim domain servers, the top-level name servers, and even the root name servers.

Everything a malicious actor needs to successfully perform an NXNSAttack is:

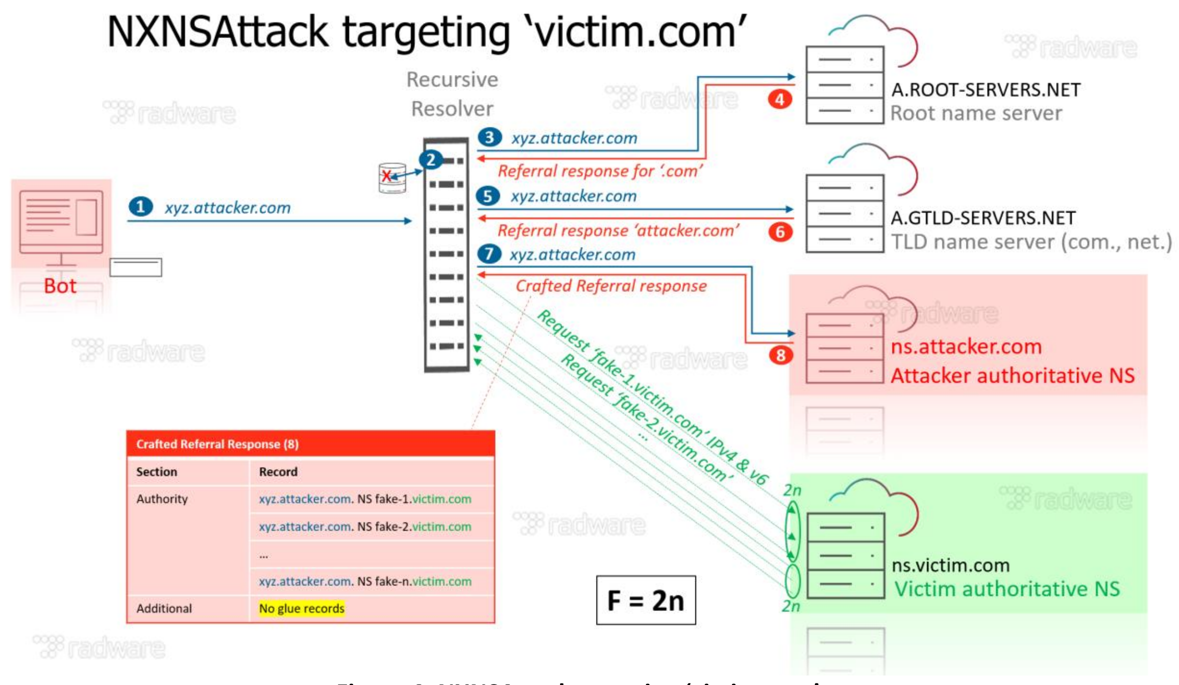
1. An authoritative name server
2. A domain name (e.g. attacker.com)
3. A client that can perform a random domain name flood

The random domain flood in requirement 3 is needed to evade local caching of resolutions by the recursive resolvers. To trigger a full recursive resolution process, the hostname in the client query should result in a cache miss. As mentioned in the introduction, Mirai provides a capable solution to perform random domain name floods at scale. Since Mirai has been in the open source domain from before the Dyn attacks, it should be clear that this list of requirements can easily be fulfilled by any pervasive attacker.

Attacking 'victim.com'

Imagine a malicious actor that owns a client or a botnet that can perform hostname resolution requests to one or more recursive resolvers. The resolvers can be private ISP resolvers or open internet resolvers. Buying a domain name such as 'attacker.com' will cost the actor anywhere from $2 to $20 annually. Renting a Virtual Private Server costs approximately $20 - $40 per month for a six-core, 8GB, 6TB data transfer flat rate multi-gigabit internet connection. More than suitable to create a capable authoritative name server for his attacker.com domain. If the actor does not own a botnet, a few small servers at $5 per month will suffice to generate enough traffic to impact a domain name, although will limit the access to private ISP resolvers and might have to rely on the open internet resolvers instead. The probability that open internet resolvers will be either protected or fixed is higher than private resolvers being protected or fixed, so a botnet still would be the preferred method.

To perform the attack, an attacker needs to configure his authoritative name server as such that any query to resolve a host in his subdomain 'attacker.com' will be replied with a referral response that contains as many random delegated hostnames as possible in the authority section. The delegated hostnames should be randomized to avoid caching by the recursive resolver. Note that negative caching on the resolver Figure 4 illustrates the attack assuming "attacker.com" is the malicious authoritative domain and "victim.com" is the targeted domain.


NXNSAttack targeting 'victim.com'
Recursive Resolver
Bot
1 xyz.attacker.com
2
3 xyz.attacker.com
4
Referral response for '.com'
5 xyz.attacker.com
Referral response 'attacker.com'
6
7 xyz.attacker.com
Crafted Referral response
8
Request 'fake-1.victim.com' IPv4 & v6
Request 'fake-2.victim.com'
...
A.ROOT-SERVERS.NET
Root name server
A.GTLD-SERVERS.NET
TLD name server (com., net.)
ns.attacker.com
Attacker authoritative NS
ns.victim.com
Victim authoritative NS
2n
2n
F = 2n

Crafted Referral Response (8)

Section	Record
Authority	xyz.attacker.com. NS fake-1.victim.com
	xyz.attacker.com. NS fake-2.victim.com
	...
	xyz.attacker.com. NS fake-n.victim.com
Additional	No glue records

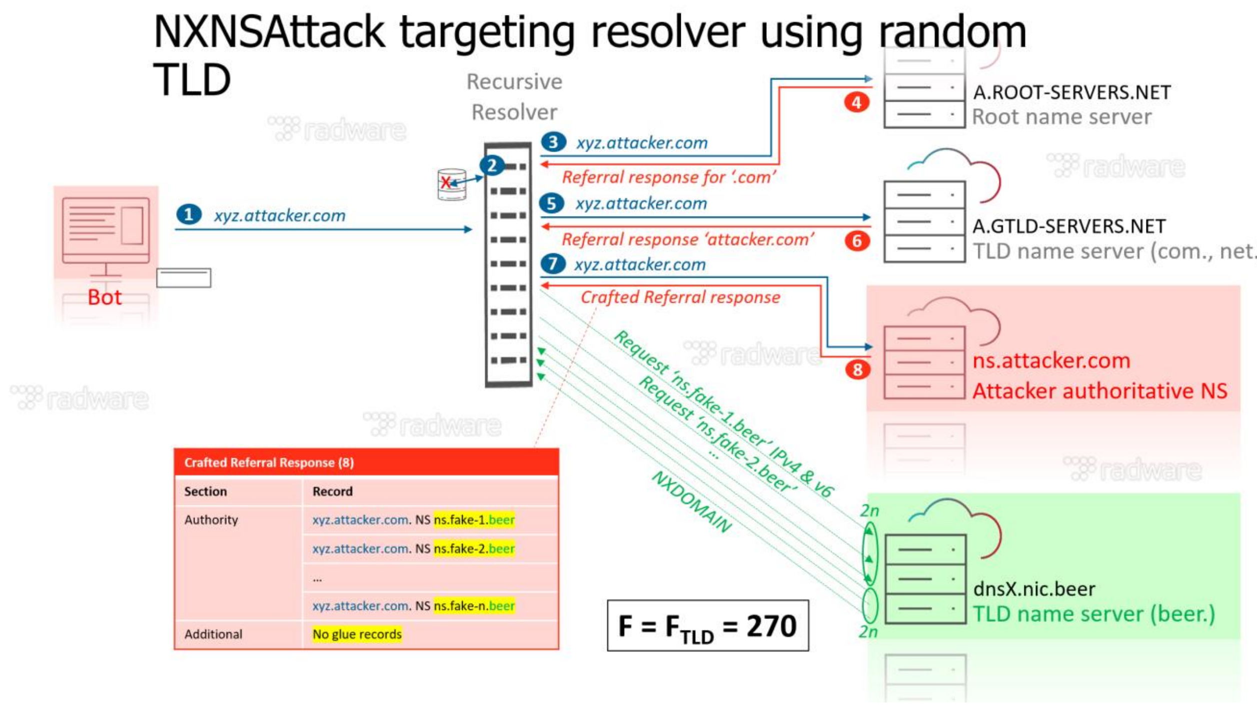
Figure 4: NXNSAttack targeting 'victim.com'

The client or bot generates a flood of random domain names under the domain ‘attacker.com’ which results in a flood of requests to the name servers of ‘victim.com’ to resolve fake delegated hostnames that were passed in the referral responses from the ‘attacker.com’ authoritative name server to the recursive resolver.

The number of requests that can be forced by the ‘attacker.com’ name server will be limited by the size of the referral response packet and as such depend on the number of characters in the random ‘attacker.com’ hostname and the randomized ‘victim.com’ delegated hostnames. The NXNSAttack paper[1] mentions a maximum of 135 hostnames based on tests conducted by the researchers.

The maximum number of recursive requests performed by a resolver can also be limited by a the server implementation. In the case of BIND, the most popular open source DNS server, a configurable parameter *max-recursion-queries* is set by default to 75, meaning that the total number of recursive requests, excluding TLD and root server requests, for a single query cannot exceed 75. Counting the recursive request to ‘attacker.com’, this means that a maximum of 74 recursive requests can be induced by ‘attacker.com’ on the ‘victim.com’ authoritative domain server. This provides a firepower of 74 requests resulting from a single client request and resulting in 1 request and 1 response to be processed by the ‘attacker.com’ server.

Because the *max-recursion-queries* parameter exempts TLD and root requests, crafting a referral response using random subdomains instead of random hostnames, one can make the recursive resolver incur the maximum of 135 delegated hostnames to resolve. This results in 2 x 135 = 270 resolution requests to the TLD name server, one IPv4 and one IPv6 request for every hostname. See Figure 5.

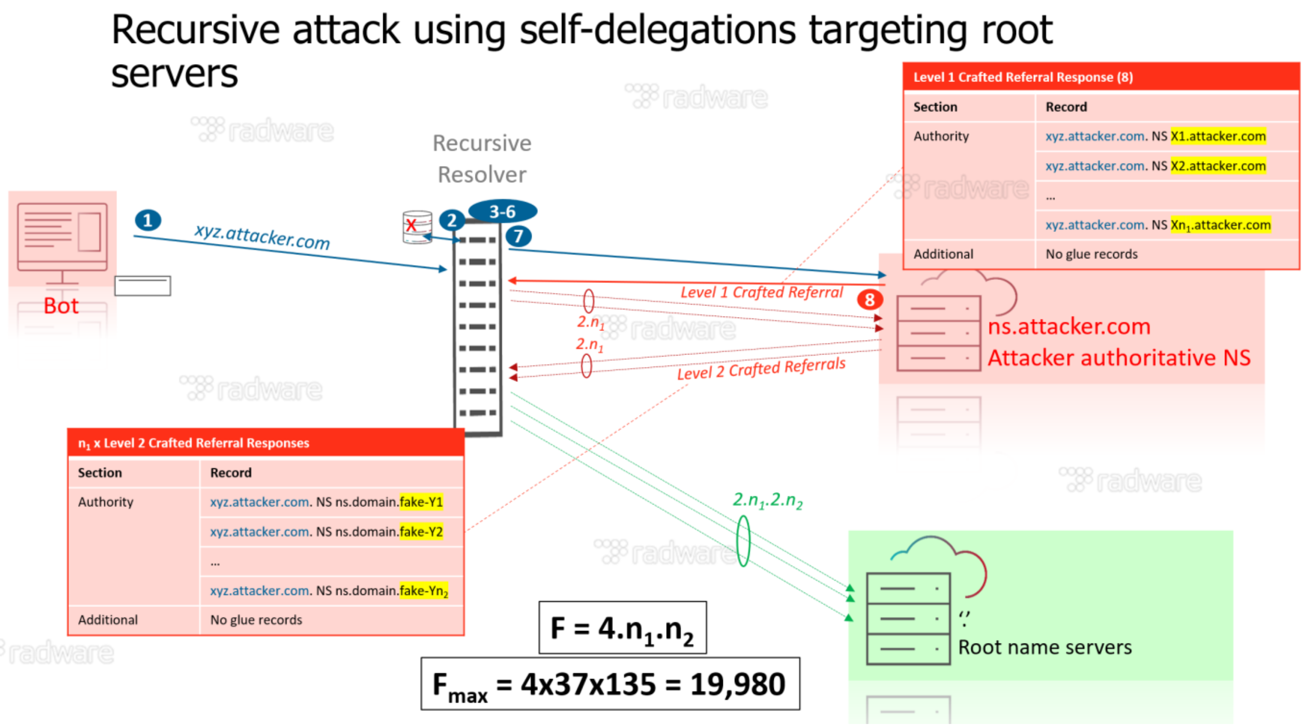


Crafted Referral Response (8)

Section	Record
Authority	xyz.attacker.com. NS ns.fake-1.beer
	xyz.attacker.com. NS ns.fake-2.beer
	...
	xyz.attacker.com. NS ns.fake-n.beer
Additional	No glue records

Figure 5: NXNSAttack targeting the resolver by using random subdomain requests to the TLD NS

Using some creative crafting of referral responses, an attacker can perform a recursive attack using self-delegations in a first level referral and delegating to another TLD in a second level referral to generate exponentially growing request streams (see Figure 6). For the request stream originating from the resolver, based on a single client request, to grow beyond the *max-recursion-queries* setting, one can ‘only’ target TLD and root name servers with this variation of the attack. The total firepower of this attack amounts to 2 x 37 first level referrals followed by 2 x 135 second level referrals for each first level referral, amounting to a total of 4 x 37 x 135 = 19,980 requests.



Level 1 Crafted Referral Response (8)

Section	Record
Authority	xyz.attacker.com. NS X1.attacker.com
	xyz.attacker.com. NS X2.attacker.com
	…
	xyz.attacker.com. NS Xn_1.attacker.com
Additional	No glue records

n_1 x Level 2 Crafted Referral Responses

Section	Record
Authority	xyz.attacker.com. NS ns.domain.fake-Y1
	xyz.attacker.com. NS ns.domain.fake-Y2
	…
	xyz.attacker.com. NS ns.domain.fake-Yn_2
Additional	No glue records

Figure 6: Recursive attack targeting root server using self-delegations

Max1Fetch Fix

As mentioned in the summary, the researchers disclosed the vulnerability in a responsible manner and provided a suggestion to solve the vulnerability. The key recursive resolver process the NXNSAttack relies on is the full recursion for every hostname in the referral response that has no glue record or is out-of-Bailiwick. Limiting the number of resolutions a resolver can do for each referral response will reduce the amplification of the attack. The researchers proposed to perform a single (Max1Fetch) or a limited (MaxNFetch) resolutions for each referral response per client query, and on subsequent queries resolve different delegation hostnames in the referrals until all have been resolved. Spreading the total number of resolutions of delegated name server hostnames for a TLD or subdomain across multiple client requests.

Protecting against NXNSAttack

To trigger the recursive resolution process, a client query should not be resolved from the recursive resolver's cache. This means that to generate an amplified request flood on the authoritative side of the resolver, a random domain name flood is required on the client side of the resolver. Eliminating the trigger will render the amplifier useless to perform attacks. As such, the same mitigation solution against Water Torture can be leveraged to protect from the NXNSAttack.

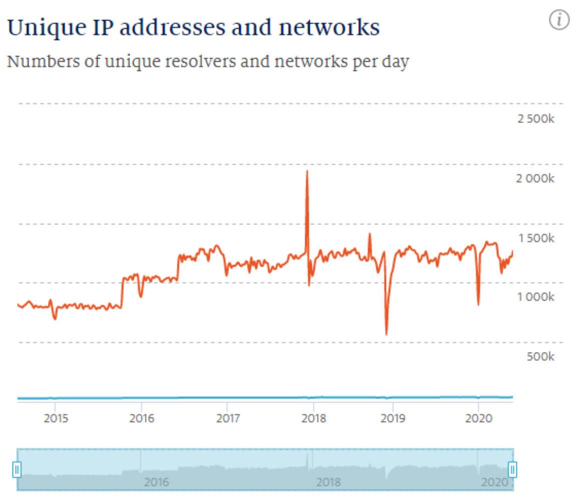
Radware DefensePro has unique capabilities to detect and protect from DNS random hostname flood attacks and will protect the recursive resolver against NXNSAttacks.

Reasons for Concern

The server fix (Max1Fetch/MaxNFetch) and the mitigation against triggering the attack from random domain floods are both to be implemented on the recursive resolver. The NXNSAttack, however, provides all the flexibility to choose the victim ranging from the recursive resolver, second level domain (SLD) name servers, top level domain (TLD) name servers, all the way up to root name servers. By consequence, authoritative domain name servers are at the mercy of the recursive resolvers to protect them from the NXNSAttack.

Knowing that any private or open resolver can be leveraged to trigger the attack against SLD, TLD, and root name servers, it becomes more concerning. A pervasive and resourceful attacker can exploit any unpatched and unprotected resolvers by gaining some clients in the realm of that private resolver through infecting connected devices with bots using known

vulnerabilities in the IoT attack surface. Radware assumes it will take a while before all recursive resolvers in the world will be patched or protected. To better appreciate the size of the problem, I refer to the Netherlands' SIDN Labs statistics[4] for the '.nl' TLD. In the past six months, their TLD name servers have been queried on average by 1 to 1.5 million unique resolvers per day. Given that '.nl' is just one of the more than 1,000 top-level domains, that puts the size of the patching problem in perspective.



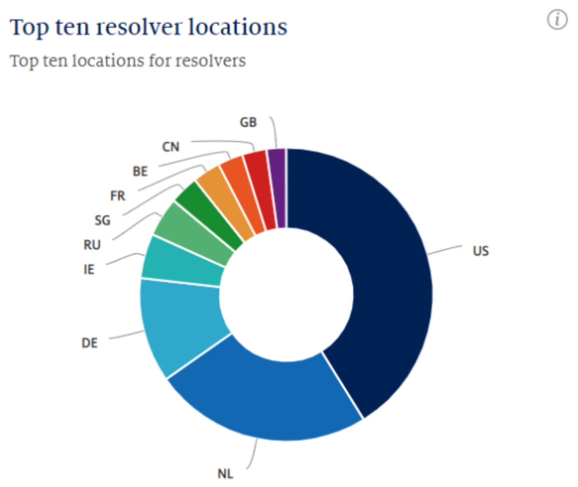


Figure 7: SIDN Labs .nl statistics

References

- NXNSAttack: http://www.nxnsattack.com/
- NXNSAttack paper: http://www.nxnsattack.com/shafir2020-nxnsattack-paper.pdf
- Radware DefensePro DNS Protection: https://www.youtube.com/watch?v=1ng6cbQATGs
- 2017's 5 Most Dangerous DDoS Attacks & Steps to Mitigate Them: https://security.radware.com/ddos-threats-attacks/ddos-attack-types/2017-most-dangerous-ddos-attacks-steps-to-mitigate/
- SIDN Labs statistics about IP address, networks and protocols used by .nl DNS resolvers: https://stats.sidnlabs.nl/en/network.html

Effective DDoS Protection Essentials

- **Hybrid DDoS Protection** - On-premise and cloud DDoS protection for real-time DDoS attack prevention that also addresses high volume attacks and protects from pipe saturation
- **Behavioral-Based Detection** - Quickly and accurately identify and block anomalies while allowing legitimate traffic through
- **Real-Time Signature Creation** - Promptly protect from unknown threats and zero-day attacks
- **A Cybersecurity Emergency Response Plan** - A dedicated emergency team of experts who have experience with Internet of Things security and handling IoT outbreaks

[4] https://stats.sidnlabs.nl/en/network.html

For further **network and application protection** measures, Radware urges companies to inspect and patch their network in order to defend against risks and threats.

Learn More at DDoS Warriors

To know more about today's attack vector landscape, understand the business impact of cyber-attacks or learn more about emerging attack types and tools visit DDoSWarriors.com. Created by Radware's Emergency Response Team (ERT), it is the ultimate resource for everything security professionals need to know about DDoS attacks and cyber security.