DNS Vulnerabilities

Johnny Rivera

Dr. Walden, Advanced Cybersecurity CIT 485

Northern Kentucky University, KY, USA

**Introduction**

While the internet was first officially implemented in 1983 with TCP/UDP features it thought very little about the future security thinking it was not going to get as popular as it is today. Now that the internet has reached its current size it is too late to go back and rebuild it around more secure communication methods. This would be too expensive and would be detrimental to many current machines which were built around the foundation of TCP/UDP protocols. The simplest answer that was found was to provide additional extensions and optional protocols that computers could implement. Domain Name Systems rely on many of these security protocols and extensions to provide security to the users accessing the information that they provide. Though they provide security it is not permanent or guaranteed thus the need for sysadmins, cybersecurity personnel, and analysist.

In this research paper I will talk about some common Domain Name System’s (DNS) vulnerabilities that cybersecurity specialist encounter. This would require going deeper into understanding how it works and why we need DNS to traverse the internet. In addition to DNS, we also talk about the dependencies that DNS has such as Transmission Communication Protocol (TCP) and the User Datagram Protocol (UDP). After introducing the foundational knowledge to DNS, I will introduce different types of DNS-based attacks. This will allow you to think like an attacker and find out the different methods they use to gain access to your sensitive data.

After introducing you to all these methods I will inform you about some prospects for DNS. While talking DNS I will also talk about what Domain Name System Security Extensions (DNSSEC) are and how they are important/implemented. During this section I will also talk about a large-scale test of DNSSEC to see how the current applications of these extensions are holding up with high load traffic [7].

Towards the end I will cover over various exploitations that can be done towards DNS through the work and testing of various pen testers (penetration testers) and researchers. This will also continue towards the last section where I mention various ways to prevent these sorts of attacks. These methods will be what some of the researchers concluded with or suggested towards some of the exploitable victims/hosts. This will inform you how secure some of these implemented extensions/add-ons can be and if they can be truly trusted.

## Domain Name System

The Domain Name System better known as DNS is a network protocol used to translate Internet Protocol (IP) addresses into more human-friendly text known as a domain name. Text such as “www.google.com” or “www.bing.com” are more user friendly unlike information like “255.255.255.255” which is a broadcast address that lets the computer know to send a message on every machine in the current network. This facilitates the use of server lookups and managing the network, it also allows companies to represent themselves when consumers try to use their products.

These server lookups are referred to as “…DNS requests, or queries, sent by devices on their network” [4]. These requests typically go to DNS clients called recursive resolvers which *resolves* these queries for the sender and sends back the answer. The process to find the answer would require “… the recursive resolver to send its own DNS queries, usually to multiple different authoritative names servers…” [4], eventually the resolver will find the authoritative name server that holds the DNS data that it needs and sent that back as a response.

While this may seem harmless from a certain point of view there are always people trying to take advantage of others when there is something to gain. Back when these protocols were invented and implemented on the internet security was not in mind. Instead of remaking the internet, which would cost a substantial amount, time, and effort not to mention the amount of systems it would affect, the better choice was to implement security protocols on top of it. Think of these as patches that would fix the road whenever there is a pothole or road deterioration, as you may be familiar with them most of time, they just go back to what they were before. Those are what these security protocols are, constantly updates and kept under watch for the potential case of attackers. The types of attacks that often affect the DNS are domain hijacking and cache poisoning.

## TCP and UDP

The Transmission Communication Protocol (TCP) is a network protocol that allows communication between two computers to send and receive packets/information. This protocol is widely known since the internet we use today relies on this protocol to function, sensing and receiving information to data centers or web-servers. It is relatively secure in the sense that it allows transfers between computers once a verification process is complete. This process is referred to as the three-way handshake, it used to be a four-way handshake but to speed up communication they combined the last two steps into one.

The User Datagram Protocol (UDP) is a network protocol that is more known as the Unreliable Protocol. This being that the main purpose of the UDP is to send the packets/information to its destination as quick as possible. The protocols only form of verification is that a destination protocol is given to be sent to. Therefore, this protocol is sometimes referred to as a connectionless transfer protocol where it is not guaranteed that the information you transfer will be all sent, it could get lost whole transferring.

The best way that many believe is that if you combine both the TCP and UDP protocol to fill in the cons that each of them have. The work of Rajkumar, K., and P. Swaminathan [9] details about this very solution. The way it would work is that UDP would transfer the Affine Cipher encrypted and compressed data over to the destination. The simplified version of the process is that the sender would also be sending the generated keys over a secure TCP connection, the receiver would just need the extract the keys and the contents using the keys from the sender. This gets the best of both these protocols and only takes a few milliseconds more than usual transferring, of course for larger files it could take a bit longer but for the bonus security that you receive from this it is a good trade.

The same work [9] also goes into detail about three different cases where an attacker would try to gain access to the data that is being transferred. One of the cases is mentions that if a cryptanalyst were to gain the generated key alone from the TCP socket it would be very difficult to try and get the original key from it. The process would require the “…length of the compressed text [to] be known from which the number of partitions and number of bits in each partition [also] has to be calculated” [9].

## Domain Hijacking

Domain hijacking is a form of DNS spoofing attack where the adversary pretends to be that server to receive victims’ information. The way that these attackers can retrieve information is that they recreate the website that the original domain server provides for their users. They then just upload their own version of it on their mimic Domain and redirect all the data to themselves, this only works if the user is inexperienced and believes that the website it just functioning differently than usual. The best way to know if the domain you are accessing has been hijacked is to look for grammatically incorrect phrases, weird fonts, and to check if the web certificate is valid. The attacker would be able to create their own certificate but the information for the server would be different. In an article published by the website Kaspersky they mention, “Once a fraudulent DNS entry is injected onto the DNS server, any IP request for the spoofed domain will result in the fake site” [6].

The most common sites that often get spoofed would be the websites that deal with sensitive data or deal with currency. This can be said that sensitive data stealing’s end goal is to obtain assets from the victim. As stated by Kaspersky, “Banking websites and popular online retailers are easily spoofed, meaning any password, credit card or personal information may be compromised” [6]. They also detail some of the actions that these attackers do with spoofing including halting security updates. This can mean that DNS spoofing can be directly tied to some ransomware, trojan or any other types of malwares if these security updates are stopped. Making sure you are checking on security updates for your client’s machine and host machines is the best way to see if you have been spoofed.

## Cache Poisoning

Cache poisoning is when instead of needing to intercept communication from the user to the server, it uses redirects to attack. It can be said that it is more user-end based than DNS hijacking. The most common scenario is that a victim will receive emails telling them that they need to perform an urgent action on a “trusted” website. If an inexperienced user clicks on a link or an ad, they get redirected to the adversary’s page instead of the original. This gets saved to the users memory from the cache as the original address to the website to substitute it. The victim’s machine will believe that the attackers IP is the original and always redirect them to the attacker’s server websites where to adversary can steal information from the user as much as they need.

While this may seem a bit less convincing than domain hijacking it does work because of the attacker’s exploitation using Social Engineering. Just like domain hijacking most of cache poisoning leads to DNS spoofing where the user gets redirected to the attacker’s server. The process is a bit different, instead of mimicking the real server cache poisoning uses malicious code to inject itself. This can be very destructive as it can be injected into a server and any machine that connects to it may be infected. As stated in the article by Kaspersky, “…cleaning an infected server does not rid a desktop or mobile device of the problem, the device will return to the spoofed site” [6]. That means that unless the victims are told or have already been stolen from, they will not know how much information is getting stolen from them.

## DNSSEC

As shown above there is many ways to manipulate DNS servers and their vulnerabilities. Thankfully the future isn’t so grim where we cannot protect our information. There is Security groups that are working on providing extensions to DNS like DNSSEC to provide forms of encryption and authentication for these DNS lookups.

Domain Name System Security Extensions (DNSSEC) uses public key authentication to protect DNS data. Public keys authentication work when there is a public key and private key pair, the DNS data owner have a private key and when someone fetch’s that data they get a public key. The public key’s purpose is to encrypt data if they wish to and the private key is used to decrypt, they are both needed if they wish to access the data. As stated in the article by ICANN, “If the signature does not validate, the resolver assumes an attack, discards the data, and returns an error to the user” [4]. This signifies that DNSSEC does have a backup measure incase someone tries to brute force their way in, this detection could provide lots of useful information as well. DNSSEC provides “data origin authentication” and “Data integrity protection” [4], these two features are all added to the resolver logic thinking.

In order to implement DNSSEC since they use cryptography you would need to also sign and validate these keys. That is the plus side to DNSSEC, they say that they provide authentication since they themselves authorize these keys. The con would be “In order for the internet to have widespread security, DNSSEC needs to be widely deployed. DNSSEC is not automatic…” [4]. This means that since DNSSEC is in its early stages of deployment the number of servers that have DNSSEC implemented are low.

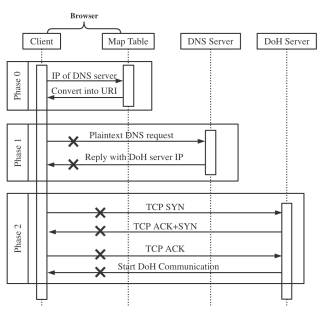
There are a lot of qualities to DNSSEC that many would want on their servers so why do we not implement it across the internet? One reason would be because it is a complicated implementation that a lot of network administrators have little knowledge in. Another reason would be that sometimes it is more destructive than productive. The work of Lian, Wilson, et al [7] detail a large-scale test that they ran on DNSSEC. They were able to gather a considerable amount of data including from queries in different regions of the world which would affect the results. To show how large it was, “Over the course of the 84 segments of our week-long experiment, we collected data from 529, 294 ad impressions, receiving DNS queries from 35,010 unique DNS resolvers” [7].

To summarize some of the data that was discussed inside the work of Lian, Wilson, et al, the percentage of fail rate to contact the resource in the ad was “0.7846%” [7] and the fail rate for resolvers with good DNSSEC configured was “1.006%” [7]. To their surprise the fail rate for “2.649%” [7] for resolvers with bad DNSSEC configured, this was theorized to be 100% since DNSSEC would reject anyone who tried to get in with bad keys for their cryptography. They go on with the statement, “…the increased security value of 2.661%. Thus, the increased security value of DNSSEC-signing a domain is relatively low, as most resolvers will not detect tampering against DNSSEC-signed domains” [7].

This does tend to question how reliable the security that DNSSEC provides is, if the number of users who try to connect to the DNSSEC enabled domain and have a valid key then there is a chance they might fail. The chances to fail to connect vs the chances to fail without even having a DNSSEC key are relatively the same then what would implementing do? This does spark the question if it is because of the load on the domain, if there were too many queries than maybe the domain was not ready for that many queries. If that’s the case than in the future with a larger load handling resolver in DNSSEC it would solve this problem but as of that article it would best to not dwindle your domain’s connectivity success rate.

## DNS Exploitations

DNS-Over-HTTPS Downgrade



The DNS-over-HTTPS (DoH) exploitation is one that mainly focuses on targeting some popular internet browsers since most of them have DoH integrated as a module. The process of DoH is explained by the work of Huang, Qing, Deliang Chang, and Zhou Li where, “DNS requests are sent in the format of an URI template…The domain name is the URI is used not only to find IP address of DoH resolver (through plaintext DNS resolution), but also to verify it’s identity (through SSL certificate verification)” [3]. This is the simplified version of what happens and by the looks of it DoH is an effective way to protect the user’s privacy.

The main problem is that an attacker can *downgrade* DoH back into plain text that DNS uses to send queries and responses. There are commonly two phases to DoH communication: “The URI resolution” [3] and the “Connection & Communication” [3]. In addition to these two phases, browsers like google chrome hard-code a mapping table so it is out of the exploitable targets. The article goes on to detail information about how a type of attacker can exploit the two phases of DoH to perform a downgrade.

Diagram

Description automatically generated

An In-path attacker, is an attacker who can inspect, intercept, and modify passing-through packets on the network. They would target phase two of the DoH communication where they would need to block TCP traffic from and to the DoH server, this would make DoH downgrade to plaintext DNS. It is much simpler than the next type of attacker but that is because in-path attackers can modify packets in the network.

An on-path attacker is one that “…can inspect the traffic of the victim and inject new packets” [3]. The on-path attacker would just need to target “…the URI resolution phase of DoH by simply blocking the specific DNS traffic sent by the victim to obtain the IP address of the DoH server” [3]. This will allow the attacker to send an unreachable IP address to pretend to be that DoH server and send response to whoever was requesting it to begin with. They could then proceed and perform DNS cache poisoning by spoofing the address they are trying to reach and make DoH downgrade back to plaintext DNS since the browser cannot reach the spoofed address.

DNS Cache Poisoning Attack Reloaded

The attack method in this has two steps which greatly heighten the bar of DNS poisoning attacks that can take place. The two crucial steps that happen in this “reloaded” DNS poisoning attack is that the attacker can use port scans by using sources ports that were previously used to initiate DNS queries at an alarming speed. This and alongside a method to extend the attack window in the connection between the attacker and the DNS server. The author of this method details that “once the source port number is known, the attacker simply injects a large number of spoofed DNS replies bruteforcing the TxIDs, which can be done in high speed, given that most servers have sufficient network bandwidth” [8].

Diagram, timeline

Description automatically generated

## Incident Response and Recovery Methods

DNS-Over-HTTPS Downgrade

The incident response that was propose by Huang, Qing, Deliang Chang, and Zhou Li’s work is that there should be revisions to DoH implementations and protocols. The implementation revision is that “…browsers that do not support strict mode should incorporate it as soon as possible” [3]. They also state that they should make those that have strict mode for their default privacy profile, in addition to that DoH should also notify the user that they are no longer connected if hey get disconnected. This would allow the user to be informed if they may become a victim of these downgrade to DNS plaintext-based attacks.

The revisions to the protocol that the authors mention is that it would be possible to replace DNS plaintext with IP addresses. This would resolve the issue where the downgrade would allow the attacker to target DNS by removing the browsers dependency of it but “…the users have to set up both URI and IP address of a DoH resolver” [3]. It would complicate it a bit more than it needs to which can leads to some problems. Another problem option that they proposed was to embed IP directly into URI template as the hostname. The problem with this would be that it “…will prevent authentication of DoH’s identity that look s into SSL certificate of its hostname” [3].

Hold-on Method

Diagram

Description automatically generated

The *Hold-on*[1] method was proposed by Duan, Haixin, et al’s article where this would give DNS stub resolvers time to verify what the real response from the recursive resolver was. This is because stub resolvers work on a first-come-first-server basis in where they do not verify that the response is valid, without *Hold-on* the stub resolver could receive injected replies if the attacker focused on speed. The best way for this to happen would be to DDoS the DNS server so that the response cannot reach there on time, the attacker’s response would go back to the stub resolver first.

The way that the stub resolver validates which response is the correct one is that they “…first learn the expected RTT and hop-count distance … associated with communication involving it’s recursive resolver, which it does using active measurement” [1]. When the responses come back they check to see which values match and are expected to be part of the response. In the image above, *Fig. 1 Hold-On while waiting for legitimate DNS reply*, the TTL matched the legitimate response and discarded the injected replies that the attacker tried to use.

Diagram

Description automatically generated

Though this may sound like a well-thought out solution the author does mention that if the attacker carefully crafts the injected packets with the correct TTL and timing with the legitimate response then the stub resolver will still receive it. That doesn’t mean all bad though, the author mentions, “However, the resolver can still detect that an attack has likely occurred (due to the differing responses)” [1]. This goes to show that this *Hold-on* method has a backup plan which can be improved on if implemented. The surprising thing is that this is only on the stub resolver side, but the author also plans to extend this method to work with recursive resolvers as well. This would greatly boost the security of resolvers on each the client and server side.

DNS Cache Poisoning Attack Reloaded

The author mentions a lot of methods to perform DNS cache poisoning by either targeting the server resolver, using the UDP against the server to simulate multiple requests until it overloads the server and lets one through. In the end they also talk about how to defend systems against this type of attack where they say, “The proposed attack is fundamentally an off-path attack and therefore can be mitigated by additional randomness and cryptographic solutions” [8]. This can bring some light to DNSSEC since they use cryptography to validate their DNS queries. Bringing that up again the load problem would still allow these connections since the servers can sometimes allow tampered with keys to go through to access data.

## Conclusion

While the internet was first officially implemented in 1983 with TCP/IP features it thought very little about the future security thinking it was not going to get as popular as it is today. Now that the internet has reached its current size it is too late to go back and rebuild it around more secure communication methods. This would be too expensive and would be detrimental to many current machines which were built around the foundation of TCP and UDP protocols. The simplest answer that was found was to provide additional extensions and optional protocols that computers could implement. Domain Name Systems rely on many of these security protocols and extensions in order to provide security to the users accessing the information that they provide. Though they provide security it is not permanent or guaranteed thus the need for sysadmins, cybersecurity personnel, and analysist.

After a few decades of operation/management the internet has seen many types of cyberwarfare. Most cases of cyber-attack’s aim were with the goal of obtaining either money or sensitive data. This paper covered most of the attacks that had these such goals and more but focusing on targeting DNS. Cache poisoning [6] also known as spoofing allowed the attackers to redirect the user data to that said attacker’s server instead of the one the user was originally trying to reach. Domain Hijacking [6] would be similar to spoofing but instead the attacker *hijacks* the original domain instead of overwriting it and can obtain information from the user trying to connect to it by adding in their malicious page/program to the webserver/server. The future of DNS can vary depending on if we continue to develop DNSSEC [4] and/or develop new DNS security extensions.

Though it may seem that there is no hope for DNS since it relies very heavily on extensions these attacks are not permanent and most of the time have dedicated teams to respond to these incidents. They develop response protocols and recovery tools to retrieve the data that was lost before it is misused. Additionally there is always new defense fortification protocols and methods being created to this day, one instance is with the proposal of *Halt* [3] where the website browser would first Halt and check if the acknowledger was who they said to be. These small but crucial contributions make browsers more secure. The challenges cybersecurity analysis and cyber defense teams face become more intricate but with the correct tools and background we can secure the future.

Bibliography

[1] Duan, Haixin, et al. "Hold-on: Protecting against on-path DNS poisoning." Proc. Workshop on Securing and Trusting Internet Names, SATIN. 2012.

[2] Harris, Brendon, and Ray Hunt. "TCP/IP security threats and attack methods." Computer communications 22.10 (1999): 885-897.

[3] Huang, Qing, Deliang Chang, and Zhou Li. "A Comprehensive Study of {DNS-over-HTTPS} Downgrade Attack." 10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20). 2020.

[4] ICANN. “DNSSEC – What Is It and Why Is It Important?” *DNSSEC – What Is It and Why Is It Important?*, ICANN, 2019, https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en.

[5] IT Sector Coordinating Council, et al. “Provide Domain Name Resolution Services and Provide Internet Routing, Access, and Connection Services Critical Functions Risk Assessment.” CISA, 2017.

[6] Kaspersky. “What Is DNS Cache Poisoning and DNS Spoofing?” *What Is DNS Cache Poisoning and DNS Spoofing?*, Kaspersky, 13 Jan. 2021, https://usa.kaspersky.com/resource-center/definitions/dns.

[7] Lian, Wilson, et al. "Measuring the Practical Impact of {DNSSEC} Deployment." 22nd USENIX Security Symposium (USENIX Security 13). 2013.

[8] Man, Keyu, et al. "Dns cache poisoning attack reloaded: Revolutions with side channels." Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020.

[9] Rajkumar, K., and P. Swaminathan. "Combining TCP and UDP for secure data transfer." Indian Journal of Science and Technology 8.S9 (2015): 285-291.

[10] Tewari, Aakanksha, and Brij B. Gupta. "Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework." Future generation computer systems 108 (2020): 909-920.