2IC80 - Kaminsky Attack

Grand Ether

Barenholz, D. (0998941), Beukers, S.C.A. (0993791), Versteeg, R. (1001652)

April 4, 2019

Click here for the video of the demo
Click here for the Github repository

Contents

Glossary

TLD	top level domain	DNS	domain name server	TTL	time to live
NS	name server	IP	internet protocol	QID	query ID

1 Introduction

Since every colloquial name for a website uses some form of DNS lookup to find the IP address, it is important that these recursive DNS services remain integral and at all times available. In 2008, Dan Kaminsky discovered that the cache of DNS servers was susceptible to being poisoned [1]. In this report we will discuss in detail what became known as the *Kaminsky attack*, or *Kaminsky exploit*, and discuss the current state of security against it.

In 2 we explain the theoretical knowledge needed to perform the Kaminsky attack. We explain the specific vulnerability which allows the Kaminsky attack to be performed. The setup used in order to perform the attack is explained in 3. A step-by-step overview of how we have achieved a successful Kaminsky attack can be found in 4. Finally, in 5 we discuss what other consequences the Kaminsky attack has yet we were unable to implement.

2 Theoretical Background

2.1 DNS Request

Whenever a DNS request is made (e.g. for `www.unixwiz.net`) from the user's PC, it follows a few basic steps.

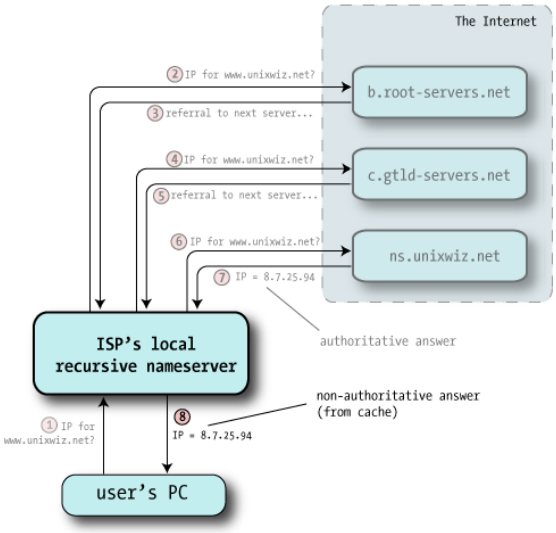


Figure 1: Basic DNS steps

1. Question: What is the IP for `www.unixwiz.net`? (1 in figure)
2. Recursive NS : I don't know, maybe `b.root-servers.net` knows, so I'll ask them. (2 in figure)
3. Authoritative NS: `www.unixwiz.net` is located at `8.7.25.94`. (7 in figure)
4. Recursive NS Answer (from cache): `www.unixwiz.net` is located at `8.7.25.94`. (8 in figure)

Note that step 2 may repeat itself several times in real life scenarios, albeit with different nameservers, such as shown in Figure 1 in steps 3, through 6.

2.2 DNS Packet

Figure 2 shows how a DNS packet looks like, including its UDP and IP headers. As can be seen in the image, the DNS data consists of following items:

Query ID (QID) The purpose of the (16-bits) query ID is identifying a question and response from other DNS traffic. This is listed in turquoise as mentioned earlier

Flags	There are multiple flags, all making up 16 bits in total. The **QR** flag denotes if the message is a question or response. The **Opcode** defines the type of DNS query (0 for a standard query). For authoritative servers replying, there is the **AA** flag standing for *Authotaritative Answer*. The **TC** flag, short for *TrunCated*, tells whether the message is truncated or not. Truncation happens when DNS message cannot fit in a single UDP datagram (e.g. when its sie is larger than 512 Bytes). The **RD** flag, short for *Recursion Desired*, expresses the host's desire to make a recursive query. The **Z** part is reserved for future use and finally the **rcode** is used in replies stating whether or not there are errors [2].
Question Count	The number of questions records in the message.
Answer Count	The number of RRs (*Resource Records*) in the answer in the message.
Authority Count	The number of authority RRs in the message.
Additional Record Count	The number of addtional information RRs in the message.
Content	The actual questions and answers, possibly with additional content. This is the message.

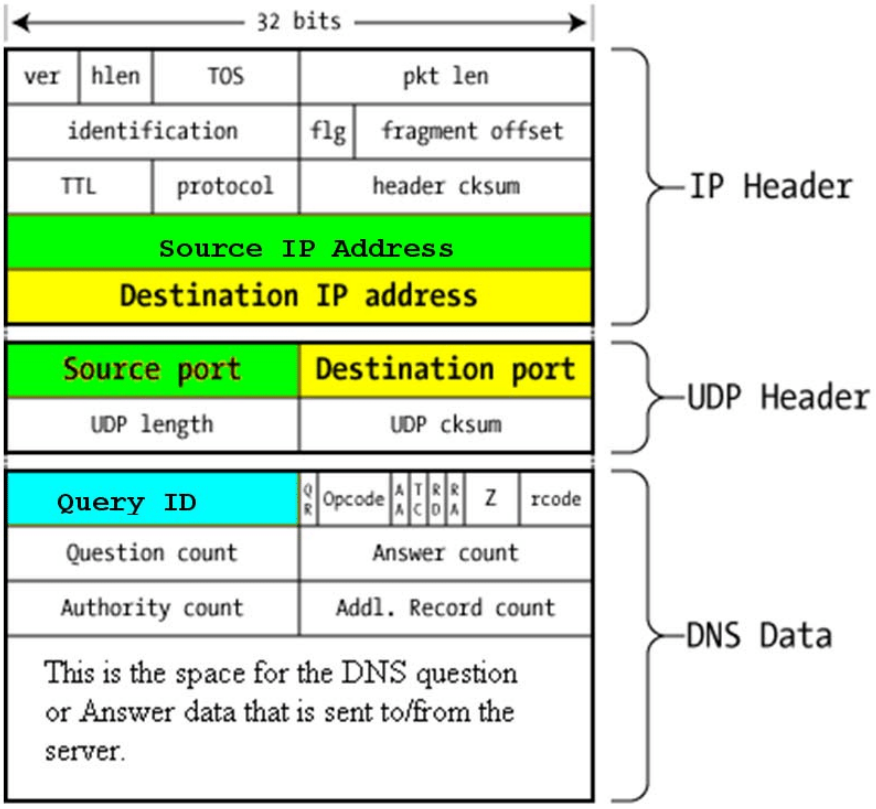


Figure 2: DNS Packet Overview

2.3 DNS cache poisoning

The idea behind DNS cache poisoning is as follows: whenever a request is made for a specific site (e.g. `www.grand-ether.com`) and it is *not* yet in the cache, then the recursive NS will ask the authoritative NS of the domain (e.g. `ns.grand-ether.com`) what the IP address is of the requested site. This query is a DNS packet with a random QID (it is assumed that the QID is random, as sequential QIDs are very insecure). The attacker can answer the query with a forged packet containing a different IP in the answer section. If the attacker is faster than the authoritative NS of the domain, then only the attacker's answer will be saved in the cache. This, however, can

only be done if the attacker can create a forged packet with the correct QID. Since the QID is merely 16 bits long, the attacker has a high chance of successfully guessing the QID when sending thousands of packets. If the attacker succeeds in guessing the QID, all sub-sequential requests for the specific site will be rerouted to the machine with the IP address as specified by the previously forged packet sent by the attacker, until the TTL has passed. Note that this site can be a malicious site, but one can also redirect traffic from a valid site to another valid site if so desired.

2.4 Kaminsky Attack

Dan Kaminsky realised that you're not limited to just one specific site when performing DNS cache poisoning. We can, for instance, forge replies from a TLD NS or even a root NS in the same manner. This means that the attacker can poison the cache in such a way that they control an entire domain, in stead of a single site. All requests for this domain (e.g. `grand-ether.com`) will go to the NS specified by the attacker. For clarity, not only can this be done for just `grand-ether.com`, it can be done on a `.com` scale as well, which can be very damaging. For now, we assume that this attack only works when the domain in question is not yet in the cache.

3 Setup

The setup consists of 4 machines as follows:

1. A TLD (top level domain) NS (name server) for `.com`, located at `192.168.56.104`.
2. An authoritative (legitimate) NS, located at `192.168.56.101`.
3. An attacker to perform the Kaminsky attack (and host its own NS), located at `192.168.56.102`.
4. A recursive NS that gets overruled by the attacker, located at `192.168.56.103`.

The setup was done using 4 VMs running a graphical installation of Debian 9 in VirtualBox. It should be noted that the setup was done in such a way that all VMs are on the same network, by setting the first network adaptor to *Internal Network*, and giving each VM the same internal network name, e.g. `intnet-gui`. Figure 3 is an illustration of the final topology used.

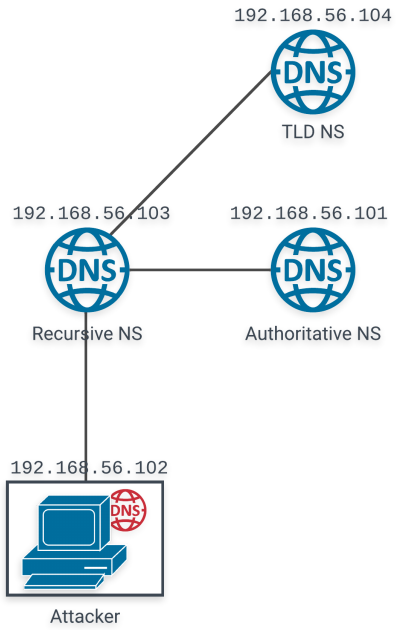


Figure 3: Network Topology

Our authoritative NSs run *CoreDNS* as DNS software, which works by means of *Corefiles* (an explanation can be found here) that read *zonefiles*. Our recursive NS runs *unbound*, which works

by means of just a configuration file. All machines have *Wireshark* installed, and the attacker machine has an installation of *scapy*. The files used for this setup can be found on our github repo which can be found here.

4 Kaminsky attack

4.1 Initialization

The discovery that Kaminsky made is already over 10 years old. This means that all proper DNS software has already been patched to prevent this attack, hence why we had to disable some features for the attack to work. Unbound originally uses a random port number for all queries, meaning that one must correctly guess the QID and port number, a total of 32 bits. It also randomly capitalizes letters of the query for extra randomness (e.g. from `example.com` to `ExAMpLe.cOM`). Due to this randomization, it is almost impossible to forge a correct packet with these features, hence we disabled random port numbers and random capitalization. A complete list of all preventive measures used by Unbound are documented at this page.

Unbound has prevented usage of the default port 53, hence we have decided to use port 1053. We use random QIDs instead of sequential QIDs, as random QIDs were more commonly used at the time this vulnerability was discovered, whereas sequential QIDs were only used by older NSs.

We have configured the zonefiles of the NSs as follows: The TLD NS is authoritative for the `.com` domain. It contains a record that `legit.com` is located at NS `ns.legit.com`, which is located at the IP address of the authoritative NS.
The authoritative NS is authoritative for the `legit.com` domain, it will return an IP address for all queries regarding this domain.
The attacker NS pretends he is authoritative for the `legit.com` domain, it will return a (malicious) IP address for all queries regarding this domain.
All files that were used for this configuration can be found here at our github repo.

We delay the network adaptor of the TLD NS, as it replies almost instantly in our setup. This is done in such a way that the attacker has enough time to send a flurry of forged packets. If not, the TLD NS replies too fast for the attacker to have a chance of guessing the correct QID. This scenario is more realistic than having no delay, as the TLD NS is never on the same local network as the recursive NS and should thus respond slower.

4.2 Initial Attack

From the machine of the attacker we will perform the Kaminsky attack on the recursive NS for the `legit.com` domain using `scapy`. We send a query to the recursive NS, asking where `legit.com` is located. If this is **not** yet in the cache, then the recursive NS will forward the query to the TLD NS. We will forge the reply of the TLD NS giving our IP address instead. To do this, we have to guess the QID of the query, which is done by sending a flurry of forged packets with different QIDs, hoping to guess the correct one. Depending on the delay of the TLD NS, we can send a different number of packets. For instance, with a delay of 100 ms, we can send 45 forged packets before the TLD NS answers, resulting in a 0.069% chance of successfully guessing the QID. If we have correctly guessed the QID in a forged packet and reply before the TLD NS, then the cache of the recursive will contain a record linking the `legit.com` domain to our IP address. Whenever a query is made to a subdomain of `legit.com`, a query will be sent to us, where we can reply with our own IP address. However, when we fail to correctly forge the reply, then we have to wipe the cache of the recursive NS to try the attack again. Note that this is not realistic, in a real environment this attack will not work anymore until the TTL of the record of the cache is over.

We have implemented this attack successfully and it works when sending a flurry of forged packets guessing the QID. We do have to reset the cache after every failed attempt. We work with a large delay of the TLD NS such that we have a higher chance to correctly guess it. To correctly guess it in one try for the demo, we will do an educated guess of the QID.

4.2.1 Packet structure

The original query package from the machine of the attacker in Scapy looks as follows:

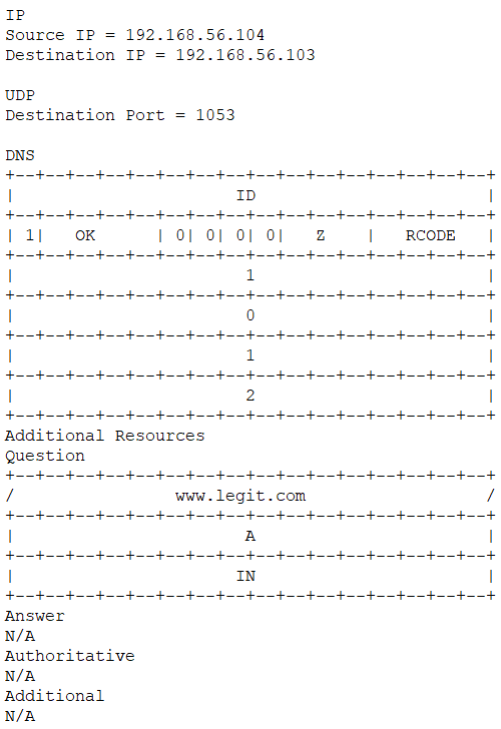


Figure 4: Figure showing the packet from the initial question packet

An example of a forged reply from the machine of the attacker in Scapy looks as follows:

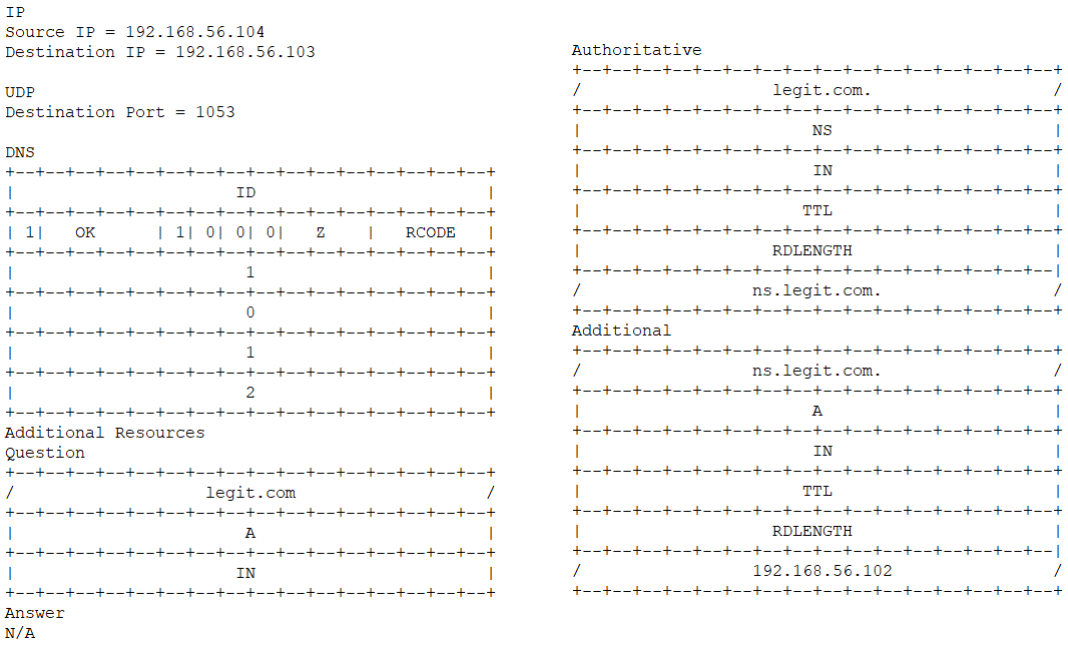
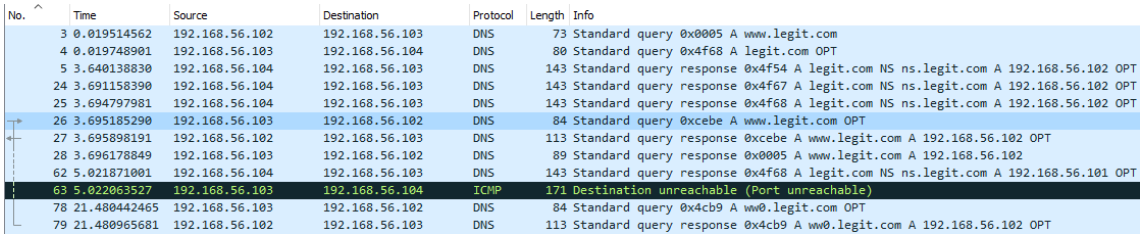


Figure 5: A figure showing the Packet used for the initial attack



No.	Time	Source	Destination	Protocol	Length	Info
3	0.019514562	192.168.56.102	192.168.56.103	DNS	73	Standard query 0x0005 A www.legit.com
4	0.019748901	192.168.56.103	192.168.56.104	DNS	80	Standard query 0x4f68 A legit.com OPT
5	3.640138830	192.168.56.104	192.168.56.103	DNS	143	Standard query response 0x4f54 A legit.com NS ns.legit.com A 192.168.56.102 OPT
24	3.691158390	192.168.56.104	192.168.56.103	DNS	143	Standard query response 0x4f67 A legit.com NS ns.legit.com A 192.168.56.102 OPT
25	3.694797981	192.168.56.104	192.168.56.103	DNS	143	Standard query response 0x4f68 A legit.com NS ns.legit.com A 192.168.56.102 OPT
26	3.695185290	192.168.56.103	192.168.56.102	DNS	84	Standard query 0xcebe A www.legit.com OPT
27	3.695898191	192.168.56.102	192.168.56.103	DNS	113	Standard query response 0xcebe A www.legit.com A 192.168.56.102 OPT
28	3.696178849	192.168.56.103	192.168.56.102	DNS	89	Standard query response 0x0005 A www.legit.com A 192.168.56.102
62	5.021871001	192.168.56.104	192.168.56.103	DNS	143	Standard query response 0x4f68 A legit.com NS ns.legit.com A 192.168.56.101 OPT
63	5.022063527	192.168.56.103	192.168.56.104	ICMP	171	Destination unreachable (Port unreachable)
78	21.480442465	192.168.56.103	192.168.56.102	DNS	84	Standard query 0x4cb9 A ww0.legit.com OPT
79	21.480965681	192.168.56.102	192.168.56.103	DNS	113	Standard query response 0x4cb9 A ww0.legit.com A 192.168.56.102 OPT

Figure 6: Figure showing the result of the Initial Kaminsky Attack working

4.3 Advanced Attack

However, the reason why the Kaminsky attack is so devastating is because the cache of the recursive NS can be overwritten in almost the same manner. When the record for the `legit.com` domain is cached, then a query for a subdomain that is not already cached is immediately sent to the NS of the domain. The attacker can forge a reply for this query, with the correct QID, saying he is authoritative for the `legit.com` domain and say that the NS of the domain is located in another IP address. When the real answer arrives, the port of the recursive NS is already closed. The cache of the recursive NS will then be overwritten and all subsequent queries for this domain will be forwarded to the new IP address.

Now we can use this information to perform the following attack from the machine of the attacker. We send a query for a subdomain of the domain record we want to replace that is definitely not in the cache. For example, one can use `ww'x'.legit.com` where x is a random number, to replace the record of `legit.com` in the cache. The recursive NS will forward the query with a new QID to the authoritative NS of `legit.com`. Now we have to reply with a forged packet, which naturally needs to have the same QID as the packet sent from the recursive NS. This packet must arrive before the reply of the legitimate authoritative NS. The forged packet specifies that we are authoritative for `legit.com` and that it is located in `ns.legit.com` using an authoritative record. It is specified that `ns.legit.com` is located at our IP address using an additional record. Note that we do **not** answer the actual query. We still have to guess the correct QID in a limited amount of time, but we can simply retry the attack after we have lost the race. This means we can keep trying until the attack is successful, when it is successful it will stay in the cache until the TTL has passed. It's been reported that success can commonly be achieved in 10 seconds [1]. This can be done on any domain (e.g. `.com`, `.net`, etc), hence why this attack is very devastating.

However, when we tried to perform the attack in our lab environment, we could only successfully overwrite the cache when we did an educated guess for the QID. If we send a big flurry of forged packets, as one would do in a real-life scenario, Unbound does **not** overwrite the cache. We believe this is due to the fact that Unbound at some point had multiple entries in the cache for the same domain name, leading to it dropping the incoming packet, but we could not confirm this.

The initial packet we send is the same as for the initial Kaminsky attack as can be seen in figure 4. The only difference is that we request the address of `ww'x'.legit.com` rather than `www.legit.com`.

The packet we use for the advanced attack can be seen in the following figure:

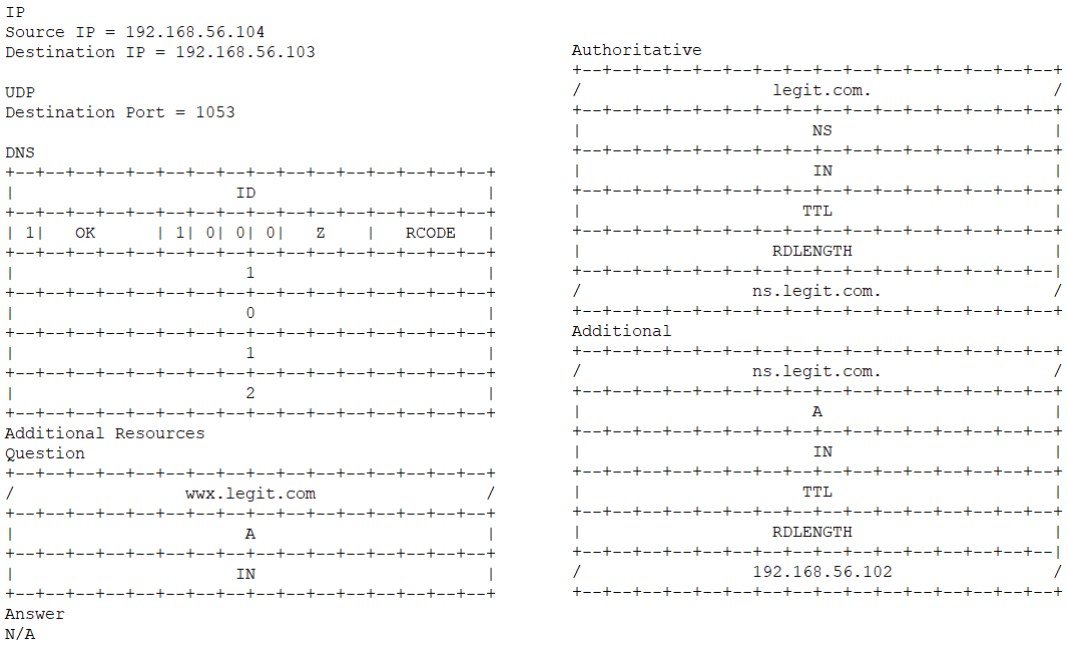


Figure 7: A figure showing the Packet used for the initial attack

No.	Time	Source	Destination	Protocol	Length	Info
3	0.023607531	192.168.56.102	192.168.56.103	DNS	73	Standard query 0x0005 A ww0.legit.com
4	0.023766919	192.168.56.103	192.168.56.101	DNS	84	Standard query 0x4ef2 A ww0.legit.com OPT
5	3.549254759	192.168.56.101	192.168.56.103	DNS	147	Standard query response 0x4ede A ww0.legit.com NS ns.legit.com A 192.168.56.102 OPT
25	3.588338775	192.168.56.101	192.168.56.103	DNS	147	Standard query response 0x4ef2 A ww0.legit.com NS ns.legit.com A 192.168.56.102 OPT
26	3.588432998	192.168.56.103	192.168.56.102	DNS	106	Standard query response 0x0005 A ww0.legit.com NS ns.legit.com A 192.168.56.102
76	5.024739378	192.168.56.101	192.168.56.103	DNS	113	Standard query response 0x4ef2 A ww0.legit.com A 192.168.56.137 OPT
77	5.024782341	192.168.56.103	192.168.56.101	ICMP	141	Destination unreachable (Port unreachable)
92	22.070910242	192.168.56.103	192.168.56.102	DNS	84	Standard query 0xedbe A ww3.legit.com OPT
93	22.071495004	192.168.56.102	192.168.56.103	DNS	113	Standard query response 0xedbe A ww3.legit.com A 192.168.56.102 OPT

Figure 8: Figure showing the result of the Advanced Kaminsky Attack working in wireshark

5 Discussion

The Kaminsky attack also has different applications that we were unable to look into, the most dangerous of these is the following. Whenever the attacker successfully replaces a record in the cache of the recursive NS, the fake website will have a wrong SSL certificate name. Perceptive users can notice this, but the majority of users will not. However, the attacker can obtain a fully-valid certificate for the target domain by hijacking the mailserver of the certificate vendor, this can be done by attacking their recursive NS via the Kaminsky attack. Even the most perceptive users cannot see that they are on a fake website. Due to time constraints and a lack of resources we were unable to look into this matter. Further research can be done in this field.

References

[1] S. Friedl, "An illustrated guide to the kaminsky dns vulnerability." http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html, 08 2008. Accessed on 2019-03-20.

[2] K. Banger, "Dns message format and name compression." http://www.keyboardbanger.com/dns-message-format-name-compression/, 09 2015. Accessed on 2019-03-22.