DNSSEC

by

Steven Dormady

An Independent Study Project

Department of Computer Science

James Madison University

Department of Computer Science

April 2026

Contents

Abstract

TBD

Introduction

1.1 Motivation

DNS by itself is insecure and vulnerable to attack. With DNSSEC, these vulnerabilities are contained and mitigated.

My personal motivation for this study was a hand on security project, proctored by Dr. Aboutabl, who has become one of my personal favorite professors here at JMU. His Information Security class helped further my interest for security, and when he mentions this study, I knew I had to seize the opportunity.

1.2 Problem Statement

1.3 Contributions

What you contribute...

1.4 VM Research

I have never dealt directly with Virtual Machines before, and building a network of them was something I had never really thought about. However, the thought excited me and it seemed like a great opportunity to learn a new skill. Setting up the first one was a bit of a

learning curve, but after setting up the first one to the specifications I wanted, I was able to clone the rest as needed.

The software I used to set up the testbed network for my DNSSEC program was VMWare Workstation. This software made it very easy to set up the virtual machines and network. Having a built in option to create a virtual network was very nice. I was also able to connect to the internet and download the necessary tools I needed, like Bind9, VMWare Tools, Wireshark, gcc and more. I also chose to use git for version control and being able to edit the code on multiple devices if needed.

1.5   VM Network

On VM Workstation, it is very easy and simple to set up a Virtual Network. Under the "Edit" tab in the menu, I was able to create a new vmnet and customize it to my needs. I made a custom network, isolated from the outside networks. The IP I used was 192.168.107.X, with a subnet mask of 255.255.255.0. The 192.168 is an IP reserved for local networks, so this was perfect for me to to use. There was an option to enable or disable DHCP service. Enabling it meant that you could let DCHP assign addresses to machines on the network. For me, I wanted to statically and manually assign IPs without and DHCP intervention, so I deselected this option to have more control in my environment.

1.6   Bind9

I installed Bind9 on all of the VMs in the Virtual Network. The version I installed was 9.18.39 (check). I decided on Bind9 because it was a good option to be used on Ubuntu machines and gave me the control I wanted.

DNS Setup

2.1 DNS Research

Firstly, I think a bit of background about DNS helps to understand why it became what it is today. DNS started as ARPANET, which was very inefficient and used a centralized hosts file to map the domain names to IP addresses. When a change was made, it had to be redistributed to all of the systems. With number of users and systems needed increasing, this became very hard to maintain, which led to DNS today.[1]

In my research about DNS, I learned a lot about what happens behind the scenes with DNS Servers. Cloudflare called it, "DNS is the phone book of the internet.", and I think this name is very fitting. [2] Like a phonebook, you look up a name (domain name) and get someone's phone number (IP address). All network systems operate with network addresses, such as IPv4 and IPv6. More or less, DNS translates domain names, like www.example.com, to IP addresses, like 134.126.126.99 (jmu.edu), so browsers can load internet resources.

The DNS naming structure is organized hierarchically in a tree-like format. Starting at the root of the tree, there is the Root Domain, which is where real world DNS begins. It delegates authority to the Top-Level Domains, or TLDs, which delegate further to Second-Level Domains, or SLDs, and continue onward, spreading out like a tree. Below in figure 2.1 is a diagram from Bind9 demonstrating this hierarchy.[1]
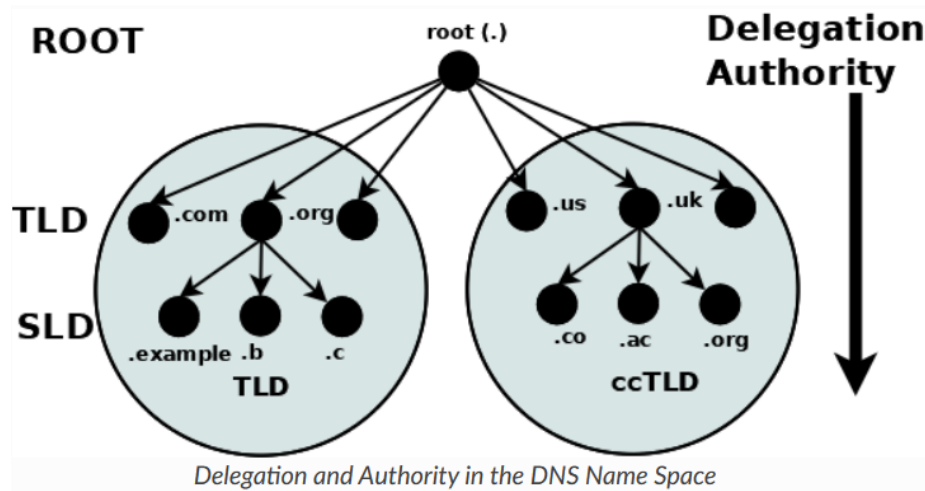
Figure 2.1: Root Delegation Hierarchy[1]

A domain is the label of a node in the tree. A domain name uniquely identifies any node in the DNS tree and is written, left to right, by combining all the domain labels (each of which are unique within their parent's zone or domain of authority), with a dot separating each component, up to the root domain. In the above diagram the following are all domain names: example.com, b.com, ac.uk, us, and org. The root has a unique label of "." (dot), which is normally omitted when it is written as a domain name, but when it is written as a Fully Qualified Domain Name (FQDN) the dot must be present.[1] As seen in figure 2.2:


```
example.com        # domain name
example.com.       # FQDN
```

Figure 2.2: FDQN example, the dot on the far right making it FQDN[1]

These root servers play a critical part of the DNS authoritative infrastructure. There are 13 root servers, which is historically tied with IPv4 data that could be packed into a 512-byte UDP message. This data limit is no longer an issue with all root servers supporting IPv4 and IPv6. In addition, almost all the root servers use anycast, with well over 300 instances of the root servers now providing service worldwide. The root servers are the starting point for all name resolution within the DNS.[1]

2.1.1 DNS Records

These DNS records are the most common records used in building a DNS server. These records are all in db files, which are database files used for each zone defined. For example, the zone ".lab" will have a db.lab file. You can name it whatever, as long as it aligns with the named.conf.local zone path defined for the zone. It is best practice to have the domain name as the db files extension. For both the A and NS records, their line begins with an @ symbol.

Start Of Authority Records

SOA records, or "Start Of Authority" records, show the domain name that the specific db file has authority over. There are several sub records within. The MNAME is the domain name of the name server of the original or primary source of data for the zone. Like a .com, or for this study, .jmu and .lab. These are typically preceded with an ns1, which will come into play later with Namserver Records later. The next record is a RNAME, which is the Responsible Person record. For this project, I have root.jmu or root.lab for the root having responsibility. The next record is the serial number, which Bind uses to load different serial versions of the zone. This record has the space to be an unsigned 32 bit number. It should be updated every time there is a change to the db file. Some developers like to use the date it was edited on to help keep track of when it was last edited. Next, the refresh record is a 32 bit time interval before the zone should be refreshed. The next record is the retry record, whish is another 32 bit time interval that would run out before a failed refresh should give up. Lastly, the expire record defines another 32 bit time value that gives an upper bound on the time interval that can run before the zone is no longer authoritative. Below is an example of an SOA record for BLANK zone.[3] An example cane be seen below in figure 2.3.

```
@       IN      SOA     ns1.lab. root.lab. (
                              5           ; Serial
                         604800           ; Refresh
                          86400           ; Retry
                        2419200           ; Expire
                         604800 )         ; Negative Cache TTL
```

Figure 2.3: An example of the SOA record in the parent servers db.lab, representing the start of authority for the zone.

Address Records

A records are address records, "A" for short. These are the records you get when a domain maps to an IP address of a given domain. This is the most fundamental type of DNS record. There are also "AAAA" records, which are for IPv6 traffic, but for this study we will only focus on IPv4 IPs. These records enable a user's device to connect with and load a website, or whatever is at the IP address.[4] As seen below, in figure 2.4, when querying for an ip at one of my DNS Servers, I am returned with the IP address of that record.

```
steve@steve-VMware-Virtual-Platform:~$ dig @192.168.107.129 computer.lab A +short
192.168.107.2
```

Figure 2.4: An example of digging for an A record, the short flag used to only give me the ip.

Below, figure 2.5, is an example of what these records look like.

```
bc        IN      A       192.168.107.1
computer          IN      A           192.168.107.2
```

Figure 2.5: An example of A records in the parent servers db.lab, computer.lab and bc.lab

Nameserver Records

NS records stand for "Nameserver" records, and indicate which DNS server is authoritative for that domain. Like in the SOA, ns1.lab shows which server is authoritative for the .lab domain. Domains can have multiple NS records, like .lab has, which indicate primary

and secondary nameservers for domains. For instance, in db.lab, there are multiple NS records, one pointing the the server itself and another pointing to the secondary server for the .jmu.lab domain.[5] See figure 2.6 for an example.

```
@          IN        NS        ns1.lab.

ns1     IN       A         192.168.107.129

jmu.lab.          IN        NS        ns1.jmu.lab.
ns1.jmu.lab.      IN        A         192.168.107.128
```

Figure 2.6: An example of NS records in the parent servers db.lab

2.1.2 The DNS Environment

There are numerous similarities between DNS with the internet and this testbed study. The bridge between the user and DNS servers is the DNS resolver or name resolution. In the project, the client digs at a server, and gets a response. This is a higher level abstraction, as there are no websites mapped to the IPs on my DNS servers, just more local IPs which could be a website. There is also no official "." or root domain, but my "root" domain is lab., which is also my top level domain. This is seen in figure 2.7. In this setup, authority is held in the parent vm and passed downward onto the child server. In real DNS, a recursive resolver is used and traverses the different levels of domains. In this study, both the parent and the child server have authority over their own domains, and you can also traverse the tree to query the parent for information on the child server, as seen below in figure 2.8.
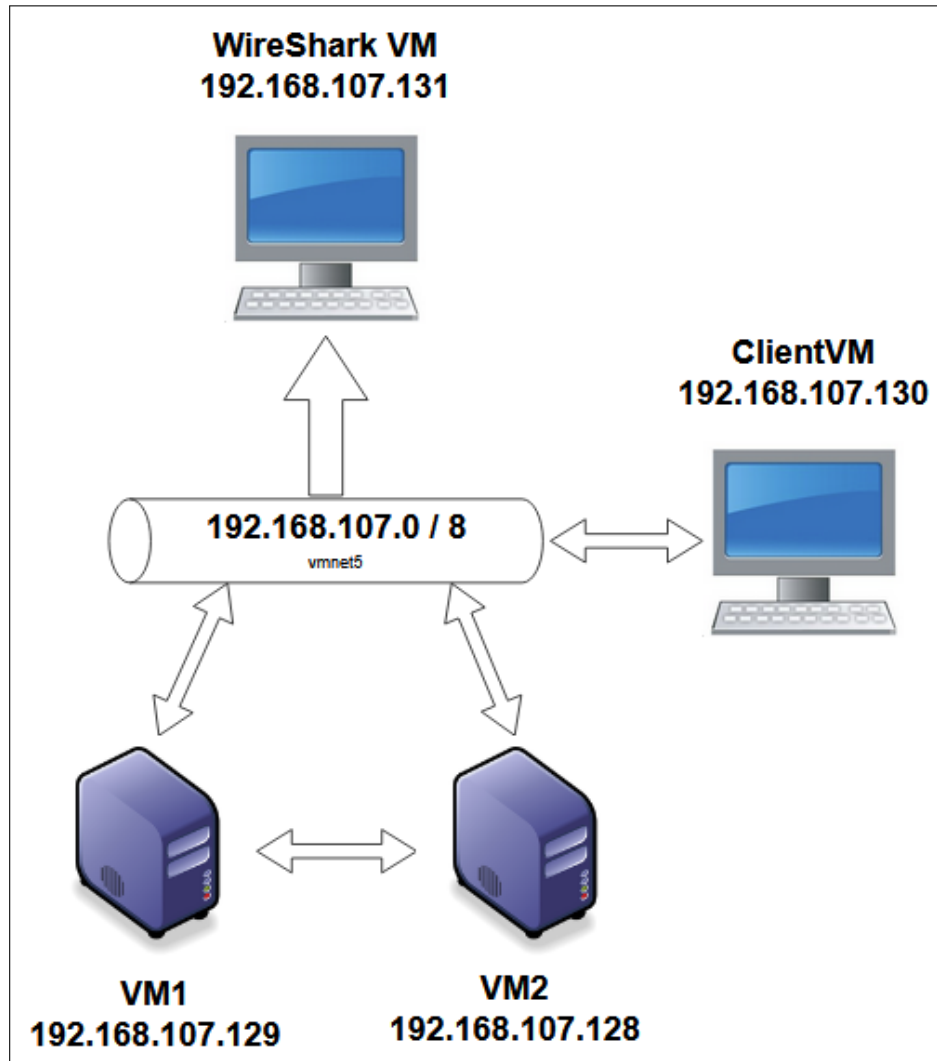
Figure 2.7: This is a diagram of the setup of the virtual environment, with inspiration of the layout coming from Dr.Aboutabl

```
;; AUTHORITY SECTION:
jmu.lab.                    86400    IN      NS       ns.jmu.lab.

;; ADDITIONAL SECTION:
ns.jmu.lab.                 86400    IN      A        192.168.107.128

;; Query time: 1 msec
;; SERVER: 192.168.107.129#53(192.168.107.129) (UDP)
;; WHEN: Tue Feb 24 22:57:16 EST 2026
;; MSG SIZE  rcvd: 102

steve@steve-VMware-Virtual-Platform:~$ dig @192.168.107.128 quad.jmu.lab A +short
192.168.107.64
```

Figure 2.8: Querying the parent for information about the child, the querying the child with that information.

8

DNS Zones

In the environment, there are different zones between the two name servers. As seen in figure 2.9, the parent zone holds .lab, with an extension to the child zone, .jmu. To be able to query the child, you need a domain name, followed by .jmu.lab. A full example of a valid query would be like this: quad.jmu.lab. Quad is the final piece, and this domain maps to 192.168.107.64. You can query the child server directly and receive responses, or you can query it through the parent server.
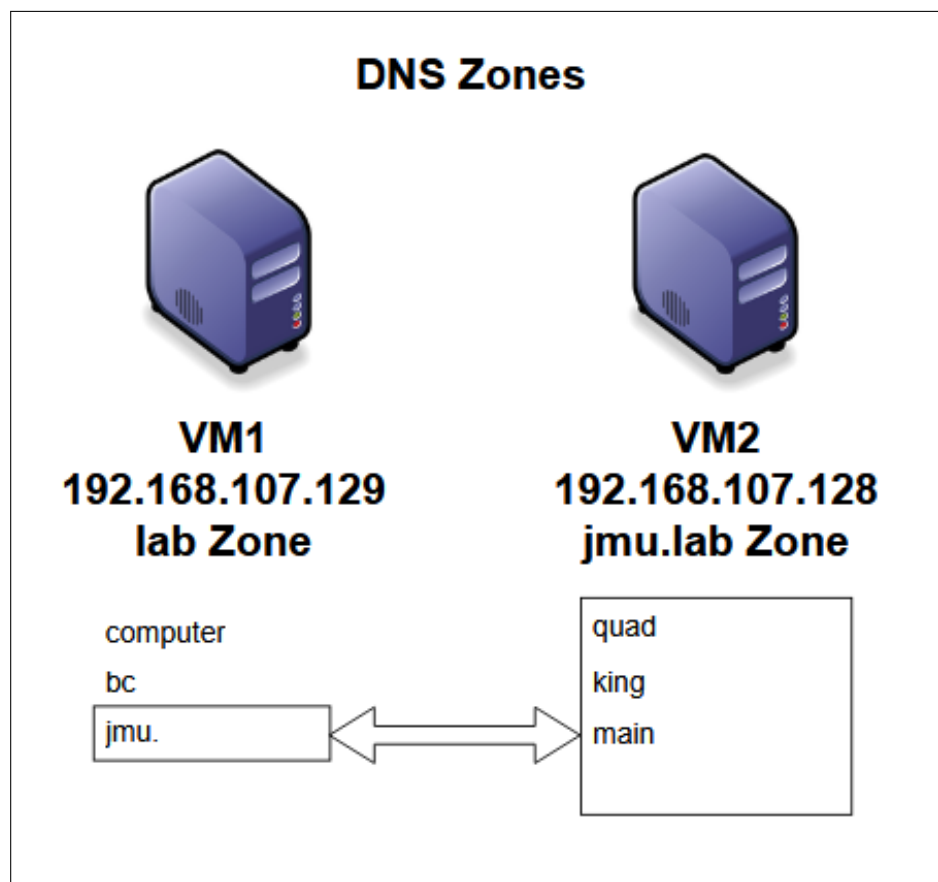

Figure 2.9: This is a diagram showing the zones and their relationship.

2.1.3 DNS Vulnerabilities

DNS without DNSSEC is very insecure and vulnerable to a plethora of attacks. Without DNSSEC, queries are readable to any person on the internet. An attacker could easily

intercept this traffic and replace it with malicious content. This is where DNSSEC comes
into play, helping to mitigate and prevent many attacks from occurring.

DNS Spoofing/Cache Poisoning

This is an especially vulnerable part of DNS without DNSSEC. This attack is a textbook
man-in-the-middle attack, when the attacked intercepts a query and replaces it with their
malicious information. DNS spoofing is when an attacker inserts their malicious address
in place of a cached address. They pretend to be a DNS nameserver and forge a reply, and
with UDP it makes it even easier for this attack to occur.[6]

DNS Hijacking

DNS Hijacking occurs when an attacker tries to redirect where the client is going with their
query.

DNSSEC Setup

3.1 DNSSEC Research

3.1.1 New DNS Records

With the regular DNS records previously discussed, there are some new records that must be taken into account.

Resource Record Signature

The Resource Record Signature, or RRSIG record, is used to verify the answers that are received.

Zone Signing Key

The Zone Signing Key, or "ZSK" for short, is the key that is used to sign data from a zone. It signs all records except for RRset that are related to keys. The ZSK has the 256 flag, indicating that it is the ZSK.

The keys follow a public key encryption scheme, with the zone keeping the private key and the

Key Signing Key

The Key Signing Key, or "KSK", is used to sign what the ZSK does not sign: key-related RRsets. This keys serves another purpose, as it is the bridge between a parent and a child zone. The flag for the KSK is 257, indicating .

Delegation Signer

The Delegation Signer Record, or DS record, is used to

3.1.2   Chain Of Trust

3.1.3   DNSSEC Options

Bibliography

[1] BIND9, "Introduction to dns and bind 9." https://bind9.readthedocs.io/en/v9.20.17/chapter1.html#the-domain-name-system-dns, 2023. Accessed: 2026-1-1.

[2] Cloudflare Learning, "What is dns?." https://www.cloudflare.com/learning/dns/what-is-dns/, 2025. Accessed: 2025-12-22.

[3] P. Mockapetris, "Domain names - implementation and specification." https://www.rfc-editor.org/rfc/rfc1035, 1987. Accessed: 2026-1-1.

[4] Cloudflare Learning, "Dns a record." https://www.cloudflare.com/learning/dns/dns-records/dns-a-record/, 2025. Accessed: 2026-1-20.

[5] Cloudflare Learning, "Dns ns record." https://www.cloudflare.com/learning/dns/dns-records/dns-ns-record/, 2025. Accessed: 2026-1-20.

[6] Cloudflare Learning, "What is dns cache poisoning? — dns spoofing." https://www.cloudflare.com/learning/dns/dns-cache-poisoning/, 2025. Accessed: 2026-2-22.

Appendix

A.1   Procedure Manual