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

Not sure what to put here, might remove.

1.3 Contributions

Is this necessary?

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 demonstrating this hierarchy.[1]

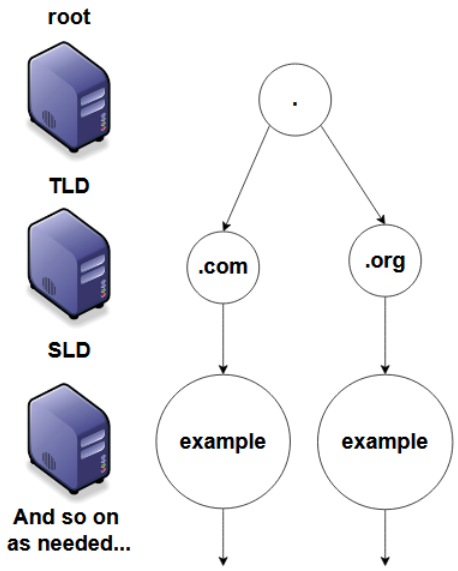

Figure 2.1: Root Delegation Hierarchy

A domain is the label of a node in the tree. A dot in a domain name means that it must traverse to another layer to get more information. Like in the above example, we start with ”example”, being at the Second-Level domain, then traverses to ”.com” at the Top-Level domain, then finally the root, ”.”, server. Put it all together and you get example.com., with the end period making the a Fully Qualified Domain Name, or FDQN. This period is usually hidden behind the scenes, and as users we do not use it. Root servers is the beginning of all DNS name resolutions, which makes the name root make sense. Like a tree, the DNS tree needs it's roots to work. However, it helps to see the root domain being reached in this instance. There are 13 root servers due to historical reasons of how many addresses could be put into a 512 byte UDP message.[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.

¡Include a db file here, need an updated pic with new terminal¿

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 2.2.

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

Figure 2.2: 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 2.3, 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 2.3: An example of digging for an A record, the short flag used to only give me the ip.

Below, figure 2.4, is an example of what these records look like.

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

Figure 2.4: 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 2.5 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 2.5: An example of NS records in the parent servers db.lab

2.2 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 2.6. 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 2.7. There are also no IPV6 records, and this is disabled in the configuration of the servers.

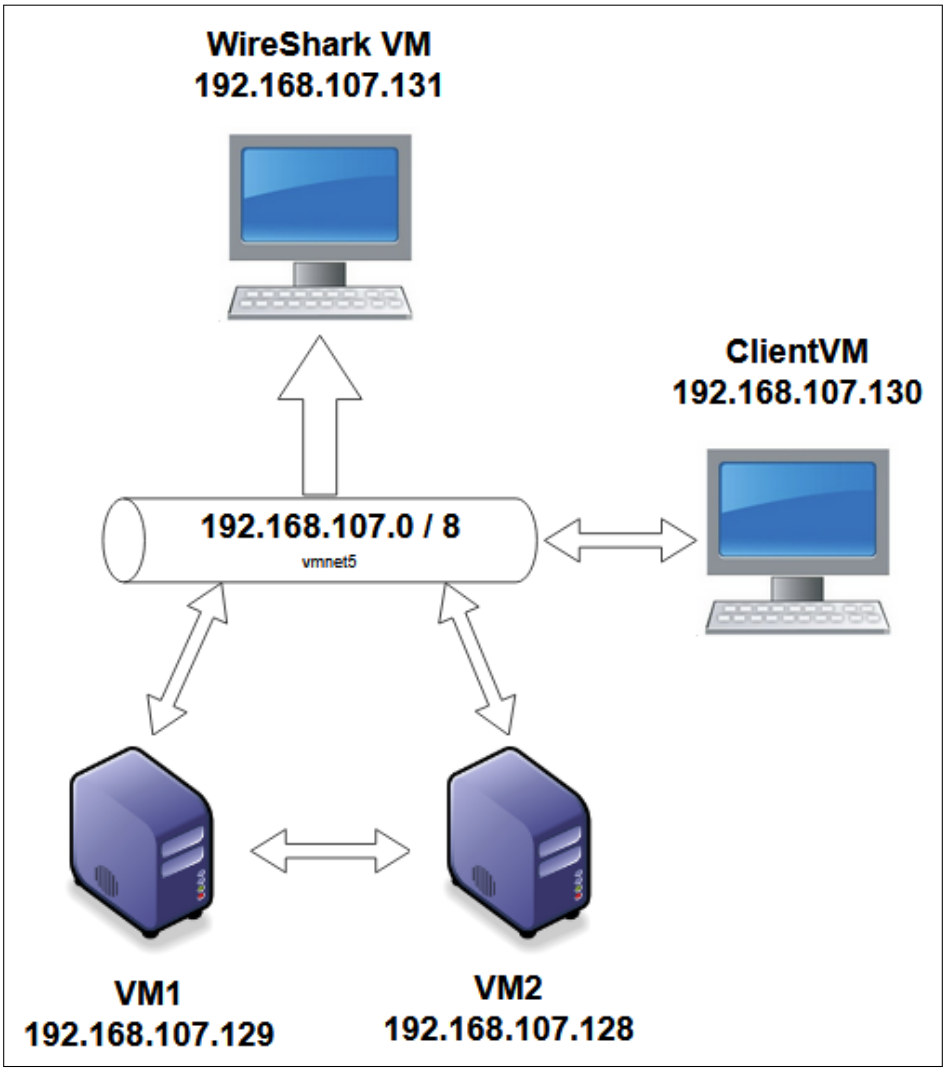

Figure 2.6: This is a diagram of the setup of the virtual environment, with inspiration of the layout coming from Dr.Aboutabl

```
;; 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 2.7: Querying the parent for information about the child, the querying the child with that information.

2.2.1 DNS Zones

In the environment, there are different zones between the two name servers. As seen in figure 2.8, 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.

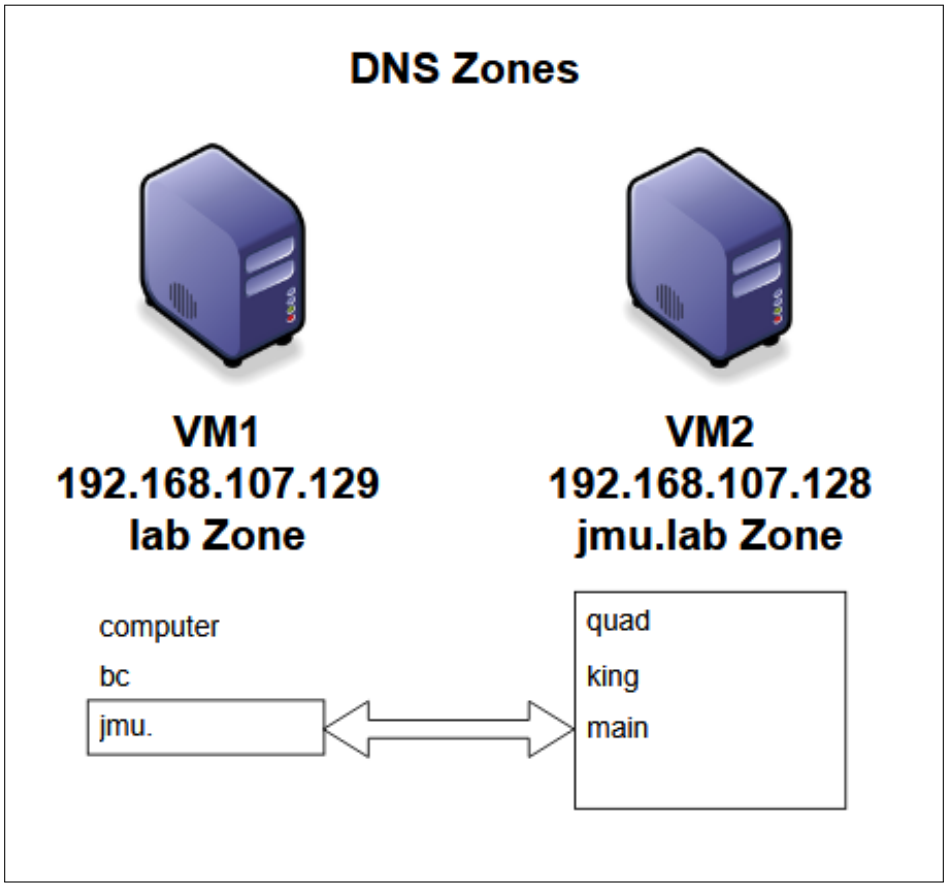

Figure 2.8: This is a diagram showing the zones and their relationship.

2.3 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. It also proves that the information is coming from a trusted, authoritative source.

2.3.1 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. This attack is using DNS Hijacking in this instance, to redirect to what the attacker wants them to see. They pretend to be a DNS nameserver and forge a reply, and with UDP it makes it even easier for this attack to occur. Cache poisoning is DNS Spoofing without having to hijack the query. The systems involved with DNS, like servers, computers, and routers can cache DNS lookups for quicker recall. An attacker can poison the cache by replacing an entry with a spoofed domain. This spoofed domain gets accessed until the cache gets refreshed and the spoofed site gets replaced.[6]

2.3.2 DNS Hijacking

DNS Hijacking occurs when an attacker tries to redirect where the client is going with their query. The can perform a man-in-the-middle attack and take the query and replace it with their IP, causing the user's machine to directly communicate with the attacker. They could also give an IP that delivers a malicious payload. These are some of the most basic types of attacks when it comes the DNS hijacking.[7]

DNSSEC Setup

3.1 DNSSEC Research

DNSSEC, or Domain Name System Security Extension, adds a layer of security to the otherwise vulnerable DNS. DNSSEC adds security by using cryptographic signatures with public key cryptography, with the private key signing it and the public key verifying. DNSSEC also implements hashing, which helps to add more confusion and making it harder for attackers to know what plain text strings were previously used.

There is an additional flag with dnssec queries, the ”ad” flag. This flag is passed along with successful DNSSEC queries, meaning ”Authenticated Data”. When you query for a domain with dnssec enabled, receiving this flag back instead of a SERVFAIL status means that the data was successfully authenticated.

Below, figure 3.1, is db.lab, but now set for DNSSEC. It changed directories to allow for permission to be more for the owner of the server (me). Originally, DNS configuration and the db files were in /etc/bind/, but now they moved to /var/lib/bind/ to allow for more freedom and not having to deal with bind ownership. Changing the zone configuration to point to the new db file allows the server to write into this directory. As you can see below, there is a good number of changes to the db file. Gone are the @ symbols preceding lines, and there is a new minimum time record. At the bottom of the figure, yuo can see a Delegation Signer record, which came from the child zone. This is an important part of the

Chain of trust, allowing the parent to verify the child is the child.[8]

```
$TTL    86400
lab.    IN      SOA     lab. root.lab. (
                              3         ; Serial
                          43200         ; Refresh
                            900         ; Retry
                        1814400         ; Expire
                           7200 )       ; minimum

lab.    IN      NS      ns.lab.
ns      IN      A       192.168.107.129

jmu     IN      NS      ns.jmu.lab.
ns.jmu  IN      A       192.168.107.128
; bc    IN      A       192.168.107.1
computer        IN      A       192.168.107.2
jmu IN DS 34939 8 2 9682051422A73107F882F98D04F47492ECA60648DC071428E43477AC7F1
42C2
```

Figure 3.1: The new DNSSEC db.lab

3.1.1 New DNS Records

With the regular DNS records previously discussed, there are some new records that must be taken into account. Regarding the Zone Signing Key and the Key Signing Key, the algorithms I used for the keys was RSASHA256, this is a good algorithm for encryption and one I am familiar with. The ZSK signs the records from the zone, and the KSK signs the ZSK. The hand shake looks a bit like this, an example of a query for computer.lab, figure 3.2.

Resource Record Set

This record is not an explicit record, rather a descriptive word to describe records that are grouped together. For instance, if I query for my parent server for DNSKEYs, I receive the RRSet that is signed, seen below in ¡figure soon¿.

Picture of DNSKEY query here

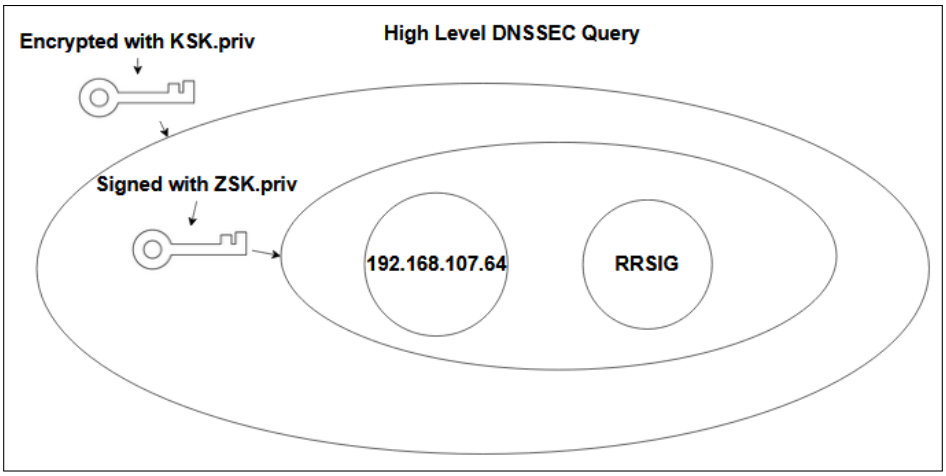

Figure 3.2: A diagram showing a high level view of a DNSSEC query

Resource Record Signature

The Resource Record Signature, or RRSIG record, is used to verify the answers that are received. These are attached to every answer that is sent with DNSSEC. The signatures are used by recursive name servers to verify that the answer to the question is valid. [9]

Diagram Showing a RRSIG

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 public being sent to the user. They use this key to then verify the signature and receive the zone data.[10]

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 that this key is the KSK. Like the Zone Key, this key also follows the public key encryption scheme. This key pair is meant to protect the ZSK for the zone being queried.

Delegation Signer

The Delegation Signer Record, or DS record, is used to ”vouch” for another zone. The parent holds a copy of this in their zone, and when asked to query about the child zone, it can verify that the response comes from the child.

3.2 Chain Of Trust

The Chain of Trust is how different levels of servers know they can trust each other. There must be trust through the whole process of the query, otherwise there is a vulnerable spot for intrusion by an attacker.[11] An example of a trust anchor is below in figure 3.3.

```
trust-anchors {
        lab. static-key 257 3 8 "AwEAAdyubupVYO97+QR5LxX3eIh8vxOI5bxfVz6ieZqMF19e
5Ul2SD+MyyY5iM1VRutYEpLmfbMDDuW3ZElJzN4BRP8huMuzRf8V4nAuB3poxIUs+kRywr4n6et46oCBF
7oWmcw792jAKKXFiuegwZ3s5NvrZufCckq5wzgaPvMxJLZLkalddzkc0KuvwAtmGsOTT7lkn3sx59JgWG
SYiL+3FNZFSZPlcDyw3QuIO1hN+Vq2NwuSR2LkOFX3miCKx4d07hr27TtP3LGJiBhcyMeJRT3i4bOBbCt
wGelA6xq5i4k1fg2+R2e+2VZtqk35GJsh8c0yqKoahz96ZCSrWCHm690=";
};
```

Figure 3.3: An Example of my Trust Anchor

3.3 DNSSEC Options

Section to talk about my dnssec policy, key management, etc.

My client is the validator, with dnssec-validation set to yes. It has a trust anchor that was manually copied from the parent zone.

3.3.1 DNSSEC Policy

I adjusted the settings in the options configuration file, mainly to specify what algorithm to use, but also to set specifics for the keys. This is seen below in figure 3.4.

```
dnssec-policy rsa-policy {
        keys {
                ksk lifetime 365d algorithm RSASHA256 2048;
                zsk lifetime 60d algorithm RSASHA256 1024;
        };
};
```

Figure 3.4: This is what I added to my dnssec policy I created called rsa-policy

3.3.2 Signing

Right now, I am using Inline Signing, which automatically generates keys and signs the files. This setting also creates and keeps a separate signed zone file.

NOTE: Still gathering information on this, and seeing if fully manual signing is worth it, because I'm not sure if you can sign RR independently

Bibliography

[1] BIND9, "Introduction to dns and bind 9." https://bind9.readthedocs.io/en/v9.18.39/chapter1.html#the-domain-name-system-dns, 2025. 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.

[7] Imperva, "What is dns hijacking / redirection attack." https://www.imperva.com/learn/application-security/dns-hijacking-redirection/, 2026. Accessed: 2026-2-26.

[8] Ubuntu Server, "Installing dns security extensions." https://documentation.ubuntu.com/server/how-to/networking/install-dnssec/, 2026. Accessed: 2026-2-20.

[9] BIND9, "Dnssec guide." https://bind9.readthedocs.io/en/v9.18.39/dnssec-guide.html, 2025. Accessed: 2026-2-25.

[10] BIND9, "Introduction to dns and bind 9." https://bind9.readthedocs.io/en/v9.18.39/chapter5.html#zone-keys, 2025. Accessed: 2026-2-25.

[11] BIND9, "Trust anchor." https://bind9.readthedocs.io/en/v9.18.39/dnssec-guide.html#trust-anchors, 2025. Accessed: 2026-2-25.

Appendix

A.1 Procedure Manual