A Demonstration of Practical DNS Attacks and

their Mitigation using DNSSEC

Israr Khan1, William Farrelly1, Kevin Curran2

1Letterkenny Institute of Technology

Port Road, Letterkenny, Co. Donegal

Email: William.farrelly@lyit.ie

2 Ulster University,

School of Computing, Engineering and Intelligent Systems,

Derry, Northern Ireland

Email: kj.curran@ulster.ac.uk

# Abstract

*The Domain Name System (DNS) is an important aspect of the Internet. It is used to map IP addresses to domain names. However, it was not designed with security in mind. DNS is vulnerable to many attacks such as Cache Poisoning and Man-in-the-Middle attack. A Cache Poisoning attack is intended to poison the cache of the server resulting in the redirection of clients to malicious domains of the attackers’ choice. The Man-in-the-Middle attack enables adversaries to steal confidential information from the servers. This is possible since DNS protocol does not provide authenticity and integrity of data. DNSSEC was proposed as a mechanism to mitigate the weaknesses within the DNS protocol. It creates digital signatures and uses keys for verification purpose which results in secure domain name resolution. The verification process takes place from the top to bottom until the domain name has been resolved securely by the authoritative name server of the client. We implement common attacks on the DNS server and demonstrate that DNSSEC is an effective solution to counter DNS security flaws. We demonstrate how to counter the Zone Transfer attack via the generation of DNSSEC keys on the name servers which prevent attackers from obtaining a full zone transfer as its request for the transfer without the keys was denied by the primary server. We also provide a detailed scenario of how DNSSEC can be used as a mechanism to protect against the attack if an attacker tried to perform Cache Poisoning. We ultimately show that a DNSSEC server will not accept responses from unauthorised entities and would only accept responses which are authenticated throughout the DNSSEC chain of trust.*

# 1. Introduction

DNS is a critical part of network and internetwork infrastructure. However, it is vulnerable, and attackers have exploited vulnerabilities within the protocol to launch various kinds of attacks against it (Gupta, 2018). The DNS protocol does not provide origin of data for authenticity and it also lacks the mechanism to provide data integrity. Taking advantage of these vulnerabilities, the attackers can forge the DNS records and direct legitimate clients to malicious domains to fulfil their own vested interests. To overcome the problems of origin authentication and data integrity, DNSSEC was proposed. It is the result of focused and continuous efforts of the security communities to secure the DNS protocol (Krishnaswamy et al., 2009). DNSSEC solves these vulnerabilities wherein security parameters are added to the DNS responses from the server which allows the client to verify that the responses originated from the intended server and that the data in the responses is not forged. Over the past decade, attacks on the Internet and private networks are on the rise. Attackers look for vulnerabilities within protocols and software, which in turn assist them in exploiting those vulnerabilities to launch attacks (Stergiou et al., 2016; Tewari & Gupta, 2017; Memos et al., 2017). Two of the most fundamental and popular attacks against the DNS protocol are Cache Poisoning and Man-in-the-Middle (MITM) attacks. In a Cache Poisoning attack, the DNS server is manipulated in a way so that it accepts and stores false data in its cache. This data does not come from an authoritative DNS server but instead it comes from a malicious user who tries to corrupt the DNS server cache by providing false information. Best practice for DNS server administrators is to randomize the UDP source port number from which caching DNS servers send out query packets as a mitigation against cache poisoning attacks. In effect, the UDP port used for a query should not be the default port 53, but instead a port randomly chosen from the entire range of UDP ports (less the reserved ports). This UDP source port randomization (SPR) makes it more difficult attackers to guess query parameters (RFC5452, 2009).

Once the DNS server cache has been corrupted, the false information will remain in the cache until the Time to Live (TTL) expires. This attack has adverse effects on the clients wanting to access the domain names from the servers. DNS data that is provided by name servers lacks support for data origin authentication and data integrity. This makes DNS vulnerable to man in the middle (MITM) attacks, as well as a range of other attacks (Ariyapperuma and Mitchell, 2007). In MITM attacks, an attacker can intercept and modify the network traffic between the resolver and the server. This occurs because the DNS protocol does not provide integrity checks and hence it is possible for the attacker to intercept and modify the data within DNS requests or responses. In 2018, a major DNS spoofing attack left the MyEtherWallet (MEW) service compromised (Nation, 2018).

The purpose of DNSSEC is to add security mechanisms such as origin authenticity and data integrity to make sure that the users can verify the origin of the data and be assured that the data has not be tempered with by anyone from its inception to reception. DNSSEC was designed to mitigate the Cache Poisoning attacks against DNS (Yu et al., 2011; Gupta, 2016). It also addresses the problems of Man-in-the-Middle attacks. DNSSEC adds digital signatures to the responses by the server and hence protects clients from deception by the malicious user who provide false data (Zou et al., 2016; Plageras et al., 2018). This is achieved by the introduction of new Resource Record (RR) types such as DNSKEY, DS, NSEC, RRSIG and other new records (Larson et al., 2005). Forensics play a significant role in determining the nature of the attacks on systems. Proper deployment and use of forensic tools can mitigate those attacks and provide a means to specify the type and degree of the attacks. Consequently, the information gained during the forensic investigation can then be used by the security experts to plan countermeasures against the attacks. (Shulman and Waidner, 2014) claim that DNSSEC is not only the most appropriate defence mechanism against Cache Poisoning attacks, but it also provides a means to carry out analysis of the attacks and delivers evidence based on the information gained. DNSSEC has been around for more than a decade, but much research has focused more on the challenges in deploying DNSSEC, the operational impacts and performance of deploying DNSSEC instead of providing practical solutions using DNSSEC as a mitigation mechanism against DNS attacks. We therefore demonstrate the implementation of practical attacks against DNS and their prevention using DNSSEC.

In this paper, we perform attacks on DNS, show mitigations using DNSSEC and determine whether the captured network data can be used as a mechanism to create a fingerprint to identify an attack. The novelty lies specially in how we show how a Zone Transfer attack can be countered with DNSSEC keys preventing attackers from obtaining a full zone transfer and how a Cache Poisoning can be prevented using the DNSSEC chain of trust. The research questions are:

RQ1: What effect does Cache Poisoning and Man-in-the-Middle attacks have on the DNS messages interchange

RQ2: How can these attacks be mitigated using DNSSEC?

RQ3: Does the captured network data provide an opportunity to create an attack fingerprint and provide a mechanism to identify the attack?

The remainder of this paper is structured as follows. Section 2 provides an overview of the Domain Name System, Section 3 provides an insight into the security flaws within DNS, section 4 provides an overview of the experimental testbed and the results while section 5 provides a conclusion to our research.

# 2. Domain Name System (DNS)

The Domain Name System (DNS) protocol is used to map domain names to corresponding IP addresses (forward lookup) and IP addresses to corresponding domain names (reverse lookup). It is a naming system that consists of hierarchically distributed DNS servers and works on the principles of client-server mechanism (Satam et al., 2015). The resolution of domain names take place hierarchically wherein referrals are made from one server to another until the queried domain name by the client has been fully resolved. The domain namespace is formed of hierarchy with distinct levels. The hierarchy starts from Root Domain followed by the Top-Level Domains also referred to as TLDs. All the TLDs are child domains of the Root Domain. Similarly, the TLDs can have several sub domains which are referred to as Second Level Domains (Chandramouli and Rose, 2013). The TLDs are categorised into two types i.e. Generic TLDs (gTLD) and Country Code TLDs (ccTLD) (Rostampour, 2012). Generic TLDs are the domains which are registered and used by organisations such as .com, .org, .net, etc. while Country Code TLDs are linked to countries such as .ie, .uk, .us, .pk. Figure 1 shows the hierarchical structure of DNS. The SLDs can also have several sub domains i.e. third level domains and so on. It is not convenient to assign a name server to each of the third level domains and other lower level domains down the hierarchy. So, to group the information relating to the primary domain of an organization, the concept of *zone* is introduced.



Figure 1: Hierarchical structure of DNS

A zone can be defined as an administrative block which is used to manage the information regarding domains. A zone may consist of a single domain or multiple domains. A zone stores the information regarding the domains in files referred to as zone files. A zone file resides on the authoritative name server, so when the name server is queried by the client for a domain, it can direct the client to the domain which it queried for. A zone file consists of the DNS data records which are called Resource Records (RR) and its associated Resource Record Types (RRtype) which specify the type of information associated with that RR (Chandramouli and Rose, 2005). The most common RR types and a description of each is given in Table 1 (Cheung and Levitt, 2000).

|  |  |
| --- | --- |
| **RRtype** | **Description** |
| A | Contains a 32-bit IPV4 address for the domain name. |
| CNAME | Canonical Name: Maps an alias to the canonical name of the domain. |
| HINFO | Host Information: Contains information regarding host. |
| MX | Mail Exchange: Contains the name of the host which acts as a mail exchange for the domain. |
| NS | Name Server: Contains the name of the authoritative name server used for the domain. |
| PTR | Pointer: contains a domain name related to an IP address. Commonly used in reverse lookup operation. |
| SOA | Start of Authority: a domain such as primary name server, Administrator email & serial number. |

Table 1: Resource Record types and their descriptions

Name servers are categorised into two types i.e. authoritative name server and caching name servers. The authoritative name server acts as an authority wherein it returns the RRs within zones when it is queried for the RRs contained within the zones. The caching name server is also referred to as a recursive name server. Such name servers provide responses by querying other authoritative name servers or from its cache. The responses from cache are provided based on the previous searches which are stored in its cache. A name server can act both as an authoritative and a caching name server (Chandramouli and Rose, 2013). In such a setting, the name server can provide authoritative responses based on the zones configured on it as well as do the job of recursive queries on behalf of the resolvers to other authoritative name servers. The DNS message is comprised of a header followed by four sections namely question, answer, authority, and additional (Cheung and Levitt, 2000). The header section of the message contains fields that are used to define the type of message. The question section contains information related to the query that is being sent to the server for resolution. The answer section contains the resource records which are used to answer the questions mentioned in the question section.

## 2.1 Zone Transfer

A zone file located in the authoritative name server consists of all the information regarding the domain including the services it provides (Pa et al., 2013). The zone file contains sensitive information and it must be kept secure so that no unauthorized entity can have access to it as it can be misused by malicious users. A DNS zone has at least one name server; however, a general rule is to have more than one name servers (Kalafut et al., 2011). The authoritative name servers are of two types i.e. primary name server and a secondary name server. These name servers are also called master name server and secondary name server respectively. The zone file within the primary name server is configured manually by the administrator of that zone while the zone file within the secondary name server is basically a replication of the data contained in the zone file of the primary server. This process of replication of zone file data from the primary name server to the secondary name server is referred to as Zone Transfer. The process of zone transfer can be done in two ways i.e. full zone transfer and incremental zone transfer. In a full zone transfer (denoted by ‘AXFR’), if an ‘AXFR’ query is issued, the authoritative name server starts a zone transfer and sends all the contents of its zone file in reply to the query. Similarly, in an incremental zone transfer (denoted by ‘IXFR’), when an ‘IXFR’ query is issued, the authoritative name server starts a zone transfer and sends the changes that occurred in the zone file since last zone transfer. Zone transfer is a very important operation of DNS and it must be carried out in a secure manner to avoid leakage of information to unauthorized people (Pa et al., 2013). The information leakage can be avoided by making sure that the primary name server allows zone transfers only to the authorised name servers.

# 3. DNS Security

The DNS protocol was not designed with security as a core function. With the passage of time, attackers exploit vulnerabilities within the protocol which allow them to implement several types of attacks on DNS. Some of the most popular attacks on DNS are Cache Poisoning, Man-in-the- Middle (MiTM), Denial of Service (DoS), Amplification attack (Aishwarya et al., 2014; Gupta, 2017). These attacks can result in corrupting the functionality of the system and have severe negative impact on the system and its users. These negative impacts may range from denying the users from getting DNS services to inserting fake records in the DNS server cache to redirect clients to illegitimate domains of the attackers’ choice. The following sections discuss and analyse the threats posed to DNS by these attacks.

## 3.1. Cache Poisoning

Cache Poisoning attack is the most common attack carried out by the adversaries against DNS. The process of storing the information in name server’s cache speeds up the process of name resolution. However, caching the information for speeding up the process comes along with the cost of introducing caching problems. In caching, the name server stores domain name information for a period which is determined by the TTL within the zone file. Until the TTL expires, the information will remain in the name server’s cache. An attacker can take advantage of this situation wherein he/she can forge the records within the zone file. The effect of this is that with the fake information provided by the attacker, a legitimate query can be referred to a malicious domain. Since the conventional DNS does not provide origin authenticity, the users of the system will assume that they are getting answers from a legitimate DNS server. The main target of the Cache Poisoning attacks are the DNS servers set up within organizations. These servers perform the function of name resolution on behalf of the clients in the organization. The Cache Poisoning attack mechanism is described in the following sequential steps and depicted in Figure 2 (Kakoi et al., 2016):

1. The client sends a query to the name server to resolve a Fully Qualified Domain Name (FQDN) to its corresponding IP address.
2. The name server will search its cache; as part of this example, it is assumed that the cache is empty, and the server sends the client’s query to the authoritative name server of domain name ‘examplesite.com’.
3. As this happens, the attacker sends massive number of camouflaged responses to the name server.
4. A vulnerable DNS server will accept the responses sent by the attacker and store it in its cache.
5. The DNS server will answer the client’s query with the fake information it accepted from the attacker. As a result, the client will be forwarded to a malicious site.



Figure 2: Cache Poisoning Process

For an attacker to succeed in such type of attack as shown in Figure 2, they are required to fake the source and destination IP addresses and port numbers. The subject of port randomization as a means of protecting the DNS against cache poisoning attacks has been approached by (Larsen and Gont, 2011) where the attackers can be prevented from knowing the port numbers by using ephemeral (short-lived) ports. Well-known port numbers for services are known to everyone, for instance, the port number for DNS is 53. Ephemeral ports are automatically chosen by the networking heap and live for a brief period. Thus, the attacker cannot be sure about the port number and must guess it to proceed with the attack. This reduces the possibility of carrying out the attack. (Dagon et al., 2008) presented a model in which they configured TTLs such that in the case where a Cache Poisoning attack fails, the poisoner (attacker) should wait TTL seconds before trying again.

## 3.2. Man-in-the-Middle (MITM)

A Man-in-the-Middle attack refers to an attack where the attacker is situated between two or more parties communicating with each other. It is a common attack and many services are vulnerable to such kinds of attack. Conventional DNS is vulnerable to MITM attacks because the query/response mechanism between the client and the server uses User Datagram Protocol (UDP); therefore an attacker can easily intercept the DNS packets and modify the information contained in those packets (Chandramouli and Rose, 2005). Moreover, the DNS messages are sent in clear text and there is no mechanism to make sure that the messages have originated from the legitimate server and that the answers from the server have not been modified in transit (Bassil et al., 2012). This weakness of the DNS protocol allows the man-in-the-middle to easily carry out malicious activities such as intercepting and modifying the messages.

This attack starts with the attacker listening to the resolver’s query to an authoritative name server to resolve query for a domain. The attacker sitting between the resolver and the authoritative name server attempts to answer the resolver’s query before the actual name server does. The answer from the attacker contains a spoofed IP address and consequently the resolver will store this fake information in its cache. The attacker has now tricked the resolver and can now provide fake information to the victim. The process is described in the steps below and depicted in Figure 3 (Rostampour, 2012):

1. The resolver asks regarding the authoritative name server for a specific domain.
2. There are two responses with regards to the resolver’s query i.e. one from the authoritative name server for the domain requested and another from the attacker. The response from the attacker contains an IP address of a fake DNS server. The attacker makes sure that the resolver receives its response first by performing a denial of service attack. The denial of service attack is aimed at slowing down the performance of the authoritative name server.
3. The requested domain is mapped to the IP address provided by the attacker.



Figure 3: MITM Attack

## 3.3 Domain Name System Security Extensions (DNSSEC)

Vulnerabilities pose serious threats to the conventional DNS; therefore, it was imperative for the security experts to provide a solution to thwart such threats. The Internet Engineering Task Force (IETF) proposed DNSSEC as one of the earliest solutions to counter DNS vulnerabilities which was first published in 1997 (Bassil et al., 2012). DNSSEC was defined by the IETF through a sequence of Requests for Comments (RFCs) aimed at improving the security of DNS (Chandramouli and Rose, 2006):

i. **RFC 4033**: Owing to the threats posed to the DNS security, this RFC defined the requirements for secure DNS (Arends et al., 2005).

ii. **RFC 4034**: The zone file specification for the DNS specification needed to be modified; therefore this RFC defined extensions to the conventional zone file and introduced new RRs to the zone file (Larson et al., 2005).

iii. **RFC 4035**: Defined extensions to support digital signatures (Arends et al., 2005b).

The DNS vulnerabilities are addressed by DNSSEC wherein it provides two security features to the messages exchanged between the resolver and the DNS server i.e. origin authenticity and data integrity. DNSSEC is an extension to the DNS protocol in which digital signatures are contained in the DNS responses to provide proof that the data received by the resolver is from an authentic server and that it has not been tempered with during the transit (Gersch and Massey, 2009). According to the DNSSEC specifications defined by IETF, the zone data must be signed. Hence, for this purpose, public key cryptography and a special set of RRs are used to authenticate the data and make sure that the data has not been altered. The new RRs associated with DNSSEC when the zone is signed are given in Table 2 (Wijngaards and Overeinder, 2009):

|  |  |
| --- | --- |
| **RRType** | **Description** |
| DNSKEY | DNS key: contains the public key used for signing the zone. This may either contain a Zone Signing Key (ZSK) or a Key Signing Key (KSK). The ZSK is used to sign the zone while the KSK is used to sign ZSK. |
| RRSIG | RRset Signature: Upon signing a zone, RRSIG is generated as the digest of the public key pair. It also specifies the validity of the signature and the owner’s name of the signing key (Lian et al., 2013) . |
| NSEC | Next Secure: Used to prove that a domain name within a zone does not exist. This is followed by pointing to next valid domain name within the zone file. This however, brings up the problem of **zone walking** where an attacker can query for non-existent domain names and walk through the zone by knowing which valid domains are present within the zone (Rose and Nakassis, 2008). |
| NSEC3 | Hashed Next Secure: It is a variant of NSEC RR. It adds obfuscation to the domain names and prevents them from zone walking by hashing the next secure domain names. |
| DS | Delegation Signer: contains digest of either a ZSK or a KSK. DS RR is used to link the signed zones for establishing the ‘chain of trust’ between them (Lian et al., 2013). |

Table 2: DNSSEC-specific RRs and their description

The name resolution process in DNSSEC starts with the client issuing a query to the name server to resolve a domain name. In case of DNSSEC, the name server is referred to as a validating name server or a validating resolver. When the validating resolvers ask the other authoritative name servers in the hierarchy for name resolution, it sends additional information (resource records) to those authoritative name servers. This is done to make sure that the authoritative name servers respond with some proof in addition to the answer (ISC,2017). Upon getting these resource records, the validating resolver will be rest assured that the answers have come from a legitimate authoritative name server.

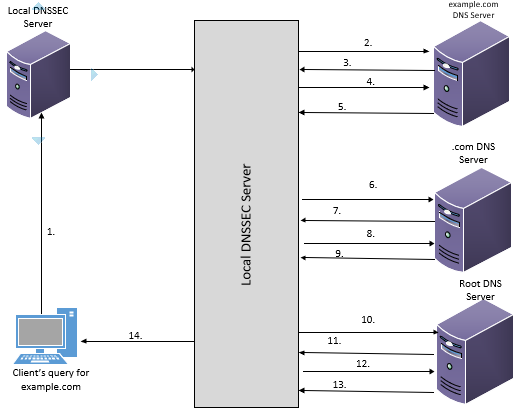


Figure 4: DNSSEC Name Resolution Process

The scenario in Figure 4 shows a step by step operation of the name resolution process in DNSSEC (ISC, 2017):

1. The client issues a query to the DNSSEC server to resolve a domain ‘example.com’
2. The local DNSSEC server receives the request and sends a query to ‘example.com’ domain wherein the query consists of a request for the A record of ‘example.com’. The query from the local DNSSEC server also has a bit set to indicate that it is looking for answers from a DNSSEC aware authoritative name server.
3. The domain ‘example.com’ is a signed domain and DNSSEC is enabled on it. Hence, the ‘example.com’ name server returns the A record as an answer. Additionally, it also returns a digital signature so that the answer can be verified by the local DNSSEC server.
4. The digital signature received by the local DNSSEC server needs to be verified. Hence, the server asks ‘example.com’ for the cryptographic keys to verify the answer.
5. The ‘example.com’ name server, in return, sends the cryptographic keys which were used to generate the digital signature which were sent in step 3. The local DNSSEC server can use these keys to verify the answers sent by ‘example.com’ name server in step 3.
6. The local DNSSEC server now turns to the ‘.com’ and requests for the verification of information which it keeps on the lower lever name server i.e. ‘example.com’.
7. The ‘.org’ name server responds to the request. When the local DNSSEC server receives this information, it compares the information to the answer received by it in step 5. If the answers in step 5 and step 7 are the same, the local DNSSEC server now has a proof that the reply from ‘example.com’ is authentic.
8. The local DNSSEC server at this point, requests the ‘.com’ name server for cryptographic keys for verification of answer which was received in step 7.
9. The ‘.com’ name server sends the cryptographic keys and the signature. The local DNSSEC server can now use these keys to verify the answer received in step 7.
10. Once the verification is done at this level, the local DNSSEC server finally turns to the root name server. It asks the root name server for the information which it keeps on ‘.com’.
11. The root name server sends information which is present on ‘.com’. The local DNSSEC server can now use this information to verify the answer received in step 9.
12. The local DNSSEC server requests for the cryptographic keys of the root name server for verification of answer it received in step 11.
13. The root name server sends its cryptographic keys and finally the local DNSSEC server can verify the answer received by it in step 11.
14. The local DNSSEC server has now retrieved all the information it needed to make sure that the answer has come from authentic sources throughout the chain. Therefore, in the last step the local DNSSEC server resolves the domain name for the client.

# 4. Experiment Results

## 4.1 Experimental Setup

In phase 1, two attacks namely Cache Poisoning and Man-in-the-Middle will be carried out on the DNS server. The DNS messages between the server and the client will be intercepted in the MITM attack for forensic investigation purposes. In phase 2, the same attacks will be carried out on the DNSSEC server. This is to identify the effect of the attacks when DNSSEC has been deployed on the DNS server (see figure 5).



Figure 5: Testing Framework

Four virtual machines were created. The first machine was created to act as the DNS master server running on Linux Ubuntu (64-bit) operating system. The second machine was created to act as the DNS slave server which was also using Ubuntu. The third machine running Ubuntu OS was created to act as the client. Kali Linux was created as the fourth machine which acts as the attacker. All the four VMs were attached to ‘NAT’ for them to be able to use the internet connection from the host machine. The promiscuous mode was set to ‘on’ in the attacker’s machine. When this mode is allowed on a device in a network, it can intercept the packets used in the communication between various devices within that network (Margaret Rouse, 2017). Hence, this mode must be allowed so that the packets sent between communicating parties can be intercepted and analysed for monitoring activities within the network. Once the network settings were completed, ‘ping’ was performed between the machines to make sure that all the four machines were interconnected. The details of all the four machines and their corresponding IPs are given in table 3.

|  |  |  |
| --- | --- | --- |
| **Machine** | **Eth0 address (Local network)** | **Eth1 address (internet)** |
| PS (Primary Server) | 10.1.100.30 | 192.168.163.151 |
| SS (Secondary Server) | 10.1.100.40 | 192.168.163.152 |
| Client | 10.1.100.60 | 192.168.163.154 |
| Attacker | 10.1.100.50 | 192.168.163.171 |

Table 3: Machines and their corresponding IP addresses

4.1.1 Servers Setup

The first step in the configuration was to install BIND and its associated packages on both the primary and the secondary DNS servers. This was achieved by using the command apt-get install bind9 bind9-doc bind9utils. After the BIND software was installed, both the servers were configured to run in IPV4 mode by adding ‘-4’ to the options in ‘/etc/default/bind9’ file. This was followed by configuration of the network interfaces file ‘/etc/network/interfaces’ to assign static addresses to both servers. IP addresses of 10.1.100.30 and 10.1.100.40 were assigned to the master and the slave servers respectively. In our lab setup, a domain called ‘testdomain.com’ has been added to the local zones file ‘/etc/bind/named.conf.local’ on the primary DNS server as well as the secondary DNS server. The zone was configured for both forward lookup and reverse lookup.

4.1.2 Client Setup

The client machine does not require a lot of configuration. This machine is used to test the name and IP address resolution for the records that are stored on the servers. Therefore, we need to add the zone name and the DNS servers to its ‘/etc/network/interfaces’ file to make sure that it queries the servers. Hence, the file is edited to include the lines dns-search testdomain.com and dns-nameservers 10.1.100.30 10.1.100.40.

4.1.3 Configuring the Cache Poisoning Attack

There are some configurations which need to be made to the primary and secondary DNS servers before the Cache Poisoning attack can be implemented successfully. For this purpose, the following configurations have been added to the servers’ /etc/bind/named.conf.options file:

• New versions of BIND DNS servers use random port numbers for DNS queries hence making it difficult to implement the attacks. However, there still are servers which use predictable/static port numbers. With regards to our attack settings, it is assumed that the port number is fixed. The port number for DNS is used here. This was achieved by editing the /etc/bind/named.conf.options file and adding the source port option query-source port 53;

• New versions of BIND DNS servers use DNSSEC option turned to ‘on’ by default to protect the server from Cache Poisoning attacks. Therefore, this option needs to be turned to ‘off’. This was achieved by editing the same file as above and adding the line dnssec-enable no.

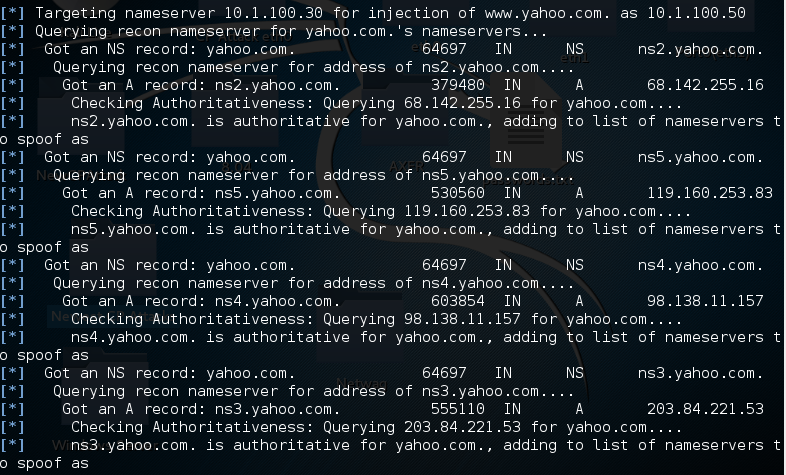
The subsections to follow demonstrate “Cache Poisoning” and “Man in the Middle” attacks and their mitigation by deploying DNS security extensions”.

## 4.2 Cache Poisoning Attack

The Cache Poisoning attack was performed by carrying out the steps as follows:

1. The network was scanned using nmapto find out the number of hosts within the network and additionally to find out the services the hosts were providing.
2. Msf Console (Metasploitable Framework Console) was opened to start the attack. This console was used to exploit the Cache Poisoning vulnerability in the server. The vulnerability was exploited by entering a series of commands in the msf console.
3. Once the exploit was executed, Wireshark was run in parallel to capture the traffic for investigation purpose.

The entire network 10.1.100.0/24 was scanned and the number of hosts scanned which were up was 4. These hosts were the virtual machines in our test environment. This was performed to find the hosts present on the network so that the attacker has a knowledge of the IP addresses used by the hosts and the services which the hosts were providing. This would enable the attacker to determine which host to attack. Upon completion of this task, it was time to launch the attack. The attacker machine was operated in the msf console mode at this point. vulnerability (auxiliary/spoof/dns/bailiwicked\_host) was exploited within the msf console. The series of steps included setting the hostname such as www.yahoo.com (the hostname which will be poisoned by giving it a false IP address). This hostname was then given a new IP address of the attackers’ choice (in this case IP address of the attacker’s machine). The source port was set to 53 because of its association with DNS. The remote host was set to the IP address of the primary name server (10.1.100.30). After all the necessary commands for the attack were filled in, the exploit was finally run at the end. The execution of the attack was started and the highlighted portions in Figure 9 show the vital points to be noted. The hightlighted portion at the top shows that the name server 10.1.100.30 has been targeted for injection of www.yahoo.com as 10.1.100.50. Different yahoo name servers started sending large amount of spoofed responses within a very short timeframe to the 10.1.100.30 server’s port 53 in an attempt to inject a poisoned record for www.yahoo.com as 10.1.100.50.



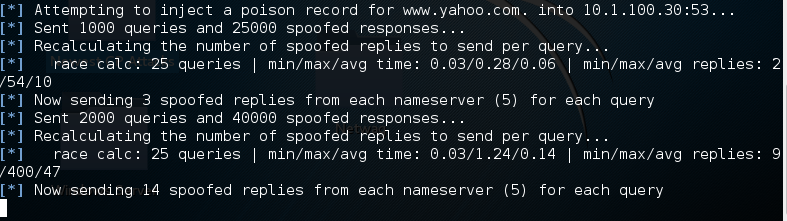


Figure 9: Attack is being executed

While the attack was being executed, the packets were captured using Wireshark. As soon as the attack started, the RECONS name server which had an IP address of 208.67.222.222 came into action. The RECONS name server’s purpose is to do a reconnaissance for the attacker machine to find out the name servers for the hostname we had set in the parameters i.e. www.yahoo.com. The RECONS name server found 5 name servers for yahoo and returned the names and IP addresses of those name servers to the attacker’s machine. The names and IP addresses of the yahoo name servers are shown in Table 4:

|  |  |
| --- | --- |
| Yahoo Name Server | IP Address |
| ns1.yahoo.com | 68.180.131.16 |
| ns2.yahoo.com | 68.142.255.16 |
| ns3.yahoo.com | 203.84.221.53 |
| ns4.yahoo.com | 98.138.11.157 |
| ns5.yahoo.com | 119.160.253.83 |

Table 4: IP addresses of yahoo name servers

In Figure 10 some sample packets have been shown which demonstrate that the RECONS name server responds with the names and IP addresses of the yahoo name servers. For brevity, one of the sample packets has been analysed. The destination IP address (192.168.163.171) is the address assigned to eth1 interface on the attacker machine which is connected to the internet. The highlighted portions indicate that the RECONS name server responded with the IP addresses of the name servers. The answer section shows the IP address for the ns1.yahoo.com name server. The remaining packets were also examined and the IP addresses for the other four name servers were noted. So now the attacker has received the IPs of the name servers with assistance from the RECONS name server.

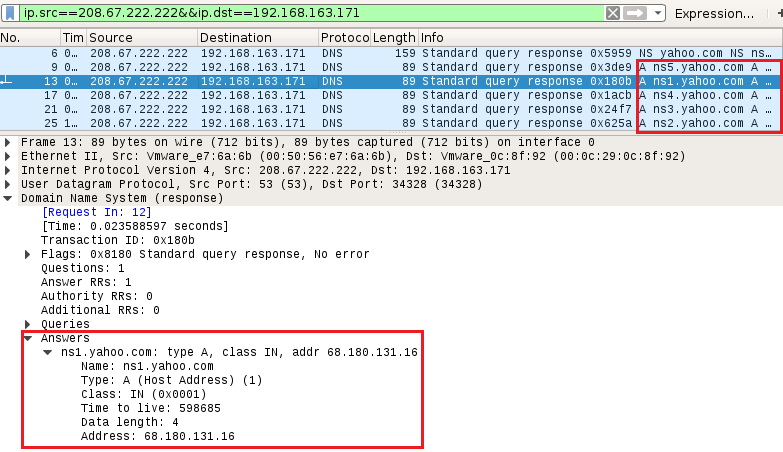


Figure 10: RECONS name server response to the attacker machine

In a Cache Poisoning attack, the attacker sends massive number of fake responses to the victim DNS server. At this stage of the attack, the attacker started sending a huge amount of fake responses to the victim DNS server. This activity was captured showing that within seconds, thousands of packets of fake responses were sent to the victim. The fake responses were sent from the IPs obtained by the RECONS name server which in turn were given to the attacker machine. Figure 11 demonstrates this activity.

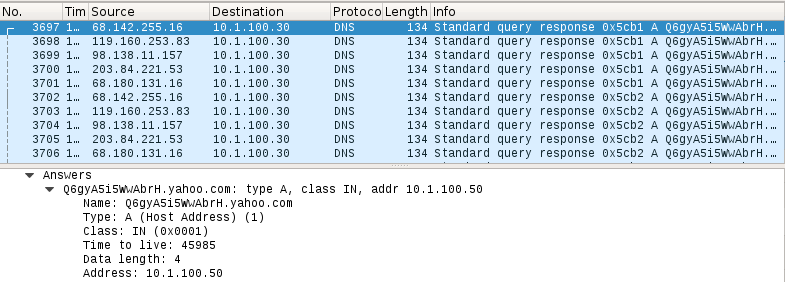


Figure 11: Spoofed responses to the victim

The packets highlighted in Figure 11 show the name servers which are continuously sending fake responses to the victim. One of the packets have been expanded to show the contents of the faked response. The answer section of the fake response to the victim DNS server shows the IP address (which in this case is the IP address of the attacker machine) to spoof the yahoo.com hostname. Even though all the steps were completed, and the attack performed in a correct manner, this attack could not poison the cache of the intended server.

## 4.3 Man-in-the-Middle Attack

The aim of the attack is to allow the attacker to perform malicious activity and attempt to gain vital information regarding the servers. The author chose to do a zone transfer attack as part of the man-in-the-middle attack. The information gained from the full zone transfer is of vital importance as it reveals sensitive information regarding the zone files in a server. The steps involved in carrying out the attack are outlined below which will be further discussed during the implementation of the attack. The following steps were taken to perform the attack:

1. The network was scanned using nmap to find out the hosts, the services they were providing and the opened ports on the hosts.
2. An SSH connection was established with the secondary DNS server.
3. The secondary server was shut down remotely from the attacker machine.
4. The IP address of the attacker machine was set to 10.1.100.40 (IP address of secondary DNS server).
5. A full zone transfer request was initiated by the attacker to the primary name server and the transfer was successful.

This section gives an insight into how the attack was practically carried out. The attack started with the attacker performing a nmap scan of the 10.1.100.0 network. The nmap scan gave the details of the hosts present on the network. It was found out that the four hosts were up. The primary server’s port 53 and secondary server’s ports 53 and 22 were open. Upon finding out the number of hosts and opened ports in the network, the attacker now had an idea of the design of the network. This gave the attacker an opportunity to plan the attack. The configurations on the primary name server only allow its secondary server to request a zone transfer and all other IPs are blocked from requesting a zone transfer. The attacker now had to find out a way to remotely shutdown the secondary server. For this purpose, the attacker established an SSH connection to the secondary server since its port 22 was open. But for the attacker to be able to establish SSH connection, it was essential to crack the password of the secondary server. For password cracking, hydra was used. A list of most frequently used passwords was downloaded and saved in a text document on the attacker machine. This was done so that a dictionary attack can be performed using hydra. Figure 6 shows how the password was cracked.

|  |  |
| --- | --- |
|  |  |
| Figure 6: Password cracked using hydra | Figure 7: Attacker logging in and shutting down the secondary server remotely |

The password cracking procedure took only 4 minutes. The next step was to remotely login to the secondary name server and shut it down. This is shown in Figure 7 wherein the attacker gets access to the secondary server machine by entering the login and the password. As soon as the attacker got access to the secondary server machine, the attacker entered the *sudo init 0* command to shut the secondary server.

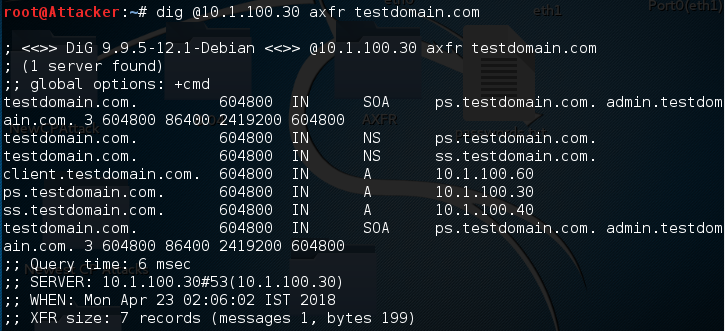


Figure 8: Successful zone transfer attack

The purpose of shutting down the secondary server remotely was to make sure that the attacker can set its own IP address to the IP address of the secondary server. If the secondary server stays up, the attacker cannot do this because an IP address duplication would occur since two machines cannot be assigned the same IP on the same network. In this way, the attacker would pretend to be the secondary server but in fact it has hijacked the IP of the secondary server. Since the secondary server was now down, the attacker changed its IP address to that of the secondary server (10.1.100.40). Once the IP address was changed, the attacker initiated a full zone transfer request to the primary server. As the attacker now has the IP address of the secondary server, therefore the primary server thinks that the secondary server is requesting a full zone transfer. Unaware of the situation, the primary server transfers its zone contents to the attacker. Figure 8 shows a successful zone transfer from the primary server to the attacker.

## 4.4 Deploying DNS Security Extensions

Since both the primary and secondary servers have been tested by the client issuing queries to the servers and the servers returning the stored information; this section of the implementation encompasses the deployment of DNSSEC on both servers. The deployment process contains mainly of performing three important steps i.e. enabling the dnssec option, generation of keys and signing of the zone with those keys. The dnssec option was disabled in the previous phase; since in this section we want to use the dnssec option, therefore it was enabled by editing the /etc/bind/named.conf.options file on both servers by adding the following lines *dnssec-enable yes* and *dnssec-validation yes.*Adding these options to the file will make sure that when the client issues query to the servers, the dnssec records will be included in the response sections.

The operation of DNSSEC requires generation of two pairs of keys namely Zone Signing Key (ZSK) and Key Signing Key (KSK). ZSK is used to sign the contents of the zone while KSK is used to sign the ZSK. The time taken to generate the keys depends upon the source which produces random numbers on the system used to generate those keys. On systems such as virtual machines, it takes longer to generate the keys. In Linux, a software package called haveged assists in minimizing the amount of time required to generate keys. It generates a stream of random numbers using the HAVEGE algorithm (Ubuntu, 2010). Once the software was installed, the keys were generated. Figure 12 & Figure 13 demonstrate the commands used for the generation of ZSK and KSK.

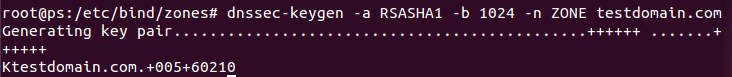


Figure 12: Generating the ZSK

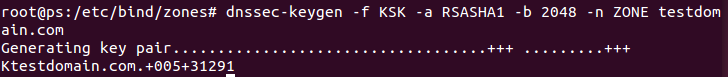


Figure 13: Generating the KSK

The commands generated 2 pairs of keys. Each pair of keys consisted of a public key and a private key. The private key of KSK key pair signs the ZSK while the private key of the ZSK key pair signs the resource records contained within a zone file. The ZSK pair was 1024-bit which was generated using RSA-SHA1 algorithms while the KSK pair was 2048-bit generated using the same algorithms. The public keys were added to the zone file. The four obtained keys were *Ktestdomain.com.+005+60210.key (public ZSK),* *Ktestdomain.com.+005+60210.private (private ZSK)*, *Ktestdomai.com.+005+31291.key (public KSK)* and *Ktestdomain.com.+005+31291.private (private KSK).*

Since the keys (which are a requirement for zone signing) were generated, it was now time to sign the zone using those keys as shown in Figure 14.

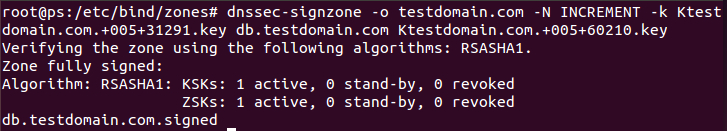


Figure 14: Signing the zone

Dnssec-signzone is a tool in BIND which is used to sign the zones. The -o parameter is used to specify the origin of the zones while the -k parameter specifies the key signing key which is used to sign the DNSKEY. The key specified at the end of the command will sign all the resource records data. The ‘increment’ part of the command incremented the zone’s serial number. The zone signing process created another zone in the zones directory i.e. *db.testdomain.com.signed*. This zone file contained the DNSKEY records which were added in the zone file earlier in addition to the RRSIG and NSEC records which were generated during the signing process. The signed zone was much larger than the original zone because of the inclusion of the DNSSEC resource records. Since the zone was now signed, it was made sure that the name server pointed to the signed version of the zone file. For this purpose, the file parameter in the ‘named.conf.local’ was changed from *db.testdomain.com* to *db.testdomain.com.signed*. This would enable the server to include the DNSSEC resource records in its response when the client issues a query to it. Our zone is now signed. The client issued a dig query to the primary name server asking it to return the contents of ‘testdomain.com’. The client included ‘+dnssec’ field in the query to indicate to the server that it is looking for dnssec answers. This is included to know whether the responding server is DNSSEC-aware. The primary name server responded with the flag set to ‘do’ which means that the server is DNSSEC-aware and is responding with DNSSEC answers. The answer section contains answers which the server sent to the client. All the answers have not been included here and only a small portion of the dig command is shown here for brevity. ‘Testdomain.com’ is the name of the domain. DNSKEY is a DNSSEC resource record which is a public key used for signing the zone. This may either be a ZSK or a KSK. 256 here indicates that this is a ZSK; 257 is used for a KSK. 3 specifies the DNSSEC protocol and 5 specifies the algorithm used to sign the zone. The value 5 here shows RSA/SHA1 was used to sign the contents of the zone. The scrambled data is the public key and 60210 is the key id.

## 4.5 Mitigation of Cache Poisoning Attack

During the demonstration of Cache Poisoning attack, our domain testdomain.com was not used by the attacker as the domain was not registered with domain registration authority. The first step would be to register the domain with an authority and choose a TLD that uses DNSSEC. The TLD would then sign the contents of the zone. This would make sure that the domain is DNSSEC enabled. Hence, in case of name resolution by the validating resolver on behalf of the client, the validating resolver can get authenticated data from the top down towards the bottom. The validating resolver can then resolve the authenticated domain name for the client. Figure 19 depicts this scenario where the client asks the validating resolver to resolve [www.testdomain.com](http://www.testdomain.com).

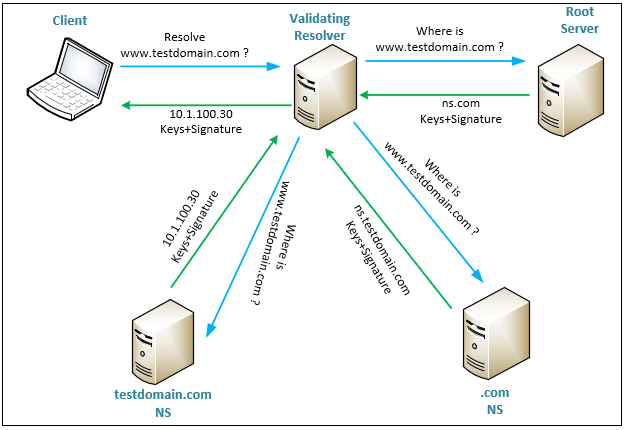
****

Figure 19: Mitigating CP Attack using DNSSEC

The validating resolver asks the Root server for the query issued to it by the client. The root server responds with the IP of the .com name server. Additionally, the Root server also sends the digital signatures to the validating server for it to verify the answers it gets. This verification process continues down the chain until the fully verified information regarding testdomain.com reaches the validating resolver. The validating resolver now resolves the domain name for the client by responding with the IP address of testdomain.com. With DNSSEC in place throughout the chain, if the attacker tries to send spoofed responses to the validating resolver, the validating resolver will not accept those spoofed responses because of the lack of verification process. Moreover, when the validating resolver refuses to accept the spoofed responses, the responses will not be stored in its cache.

## 4.6 Mitigation of Man-in-the-Middle Attack

In the previous phase, a zone transfer attack was performed as part of the man-in-the-middle attack. This section demonstrates as to how the zone transfer attack could be mitigated using DNSSEC. The mitigation process involves securing the zone transfer between the DNS servers by generating DNSSEC keys and using those keys to perform the zone transfer activity. The first step involved is generation of keys using ‘dnssec-keygen’ tool as shown in Figure 15.



Figure 15: Generating dnssec keys

The command shown in the generated a 128-bit key pair i.e. Ktkey.+157+075975.key and Ktkey.+157+07975.private using HMAC-MD5 algorithm. The HOST part of the command indicates that we intend to generate keys for the host rather than a zone. The contents of the private key file contained the key which would be used for secure zone transfer. The idea is to restrict the zone transfer without the key. The generated key was configured on both the primary and secondary servers. Once all the configurations were completed, a full transfer request without the key was made from the secondary server to the primary server. The full zone transfer request was denied by the primary server since the key was not included in the request by the secondary server. This can be seen in Figure 16

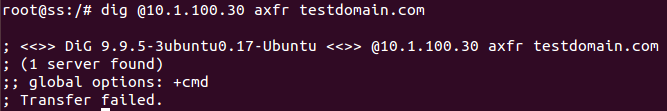


Figure 16: Zone transfer request without the key

Next, a full zone transfer request was made using the key. This time the request was fulfilled, and full zone transfer was completed successfully. Figure 17 shows a snippet of the successful secure zone transfer. The long string in the command is the key and ‘transfer’ is the name given to the key.



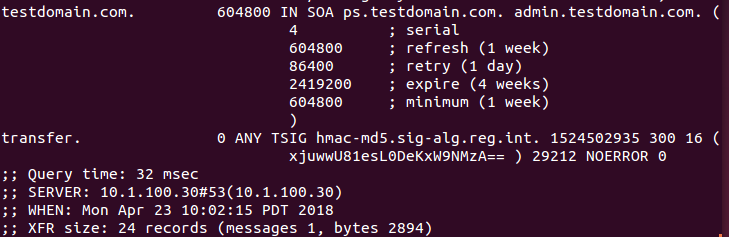


Figure 17: Successful zone transfer using the key

The zone transfer network packets process were captured. The secondary server requests a full transfer of all the contents in the zone and the primary server responds to the secondary server by transferring all the contents within the zone.With regards to the man-in-the-middle attack in the previous phase, the attacker was permitted a full zone transfer because the attacker had brought down the secondary server and used its IP for the zone transfer request. In the current scenario, the secure zone transfer provides an extra layer of security and even if the attacker manages to use the IP of the secondary server for the zone transfer, the primary server would not allow it. The attacker needs to have the key to be able to obtain a full zone transfer. The secondary server may choose to keep the key in a safe place so that no unauthorised person can have access to that key. To demonstrate how the attacker gets prevented from obtaining a full zone transfer, the attacker repeated the entire process of obtaining access to the secondary server. Once the attacker got access, the attacker shut down the secondary server and initiated the full zone transfer request to the primary server. The attacker was denied a full zone transfer as can be seen in

Figure 18.

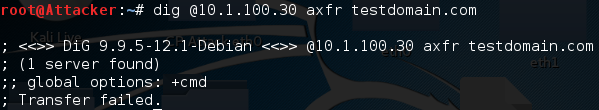
****

Figure 18: Attacker prevented from obtaining contents of the zone

# 5. Conclusion

A testing framework for the two attacks was provided. The first attack looked at how an attacker can retrieve sensitive information from a vulnerable DNS server. This was showcased by implementing a zone transfer attack in which the attacker managed to retrieve the contents of the zone file residing within the name server. The consequences of this attack can be cumbersome as the DNS server contains information regarding the domains which should not be disclosed to unauthorised users. The retrieved information may reveal the layout of the network to the attacker; which can be used to launch attacks such as poisoning or spoofing the server. Using the Cache Poisoning vulnerability against a DNS server, the attackers can send malicious information to the server and cause the server to store that fake information in its cache. The attack was demonstrated in which the attacker used two methods to implement the attack. The first showed how an attacker can add fake entries to the compromised client’s host file. The second method saw the attacker send massive amounts of fake responses to the server resulting in the server accepting those fake responses and storing them in its cache. It was concluded that the Cache Poisoning vulnerability can exploit DNS and hence compromise the security of the DNS servers.

To counter the Zone Transfer attack, DNSSEC keys were generated. Generation of DNSSEC keys on the primary and secondary name server for authentication purpose prevented the attacker from obtaining a full zone transfer as its request for the transfer without the keys was denied by the primary server. The Cache Poisoning attack was not performed on the domain which was configured on the primary name server. Rather, the attack was performed against a random domain to show how the attack works. However, a detailed scenario was provided detailing how DNSSEC can be used as a mechanism to protect against the attack if an attacker tried to perform Cache Poisoning against our domain. The scenario demonstrated that the DNSSEC server would not accept responses from unauthorised entities and would only accept responses which are authenticated throughout the DNSSEC chain of trust.

During the execution of the attacks, network packet inspection was performed where traces of the attack pattern could be clearly seen. The captured packets indicated that a huge amount of fake responses were sent by the name servers which contained the spoofed address for the domain. Although the DNS protocol plays a significant part and acts as the backbone in providing the name resolution services to the clients on the internet and private networks, it does not provide security owing to the lack of authenticity and data integrity in its design. The adversaries can take advantage of these vulnerabilities by exploiting the weaknesses in the protocol and in turn causing a serious threat to the services provided by DNS. Keeping in view the weaknesses of DNS protocol, the IETF designed DNSSEC which introduces the use of public cryptography and a set of new resource records for authenticating the responses from the authoritative name servers. The use of keys and signatures make sure that the information received by the client is originating from authentic sources and that the data has not been tempered with in transit.

# References

Aishwarya, C., A, R.M., Hosmani, S., Sannidhan, M.S., Rajendran, B., Chandrasekaran, K., Bindhumadhava, B.S., 2015. DANE: An inbuilt security extension, in: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT). pp. 1571–1576. https://doi.org/10.1109/ICGCIoT.2015.7380717

Arends, Roy, Austein, Rob, Larson, Matt, Rose, Scott, 2005a. RFC 4033 - DNS Security Introduction and Requirements [Online]. URL https://datatracker.ietf.org/doc/rfc4033/?include\_text=1 (accessed 10.30.17).

Arends, Roy, Austein, Rob, Larson, Matt, Rose, Scott, 2005b. RFC 4035 - Protocol Modifications for the DNS Security Extensions [Online]. URL https://datatracker.ietf.org/doc/rfc4035/?include\_text=1 (accessed 10.30.17).

Ariyapperuma, S. and Mitchell, C. (2007) Security vulnerabilities in DNS and DNSSEC. The Second International Conference on Availability, Reliability and Security (ARES'07),NY, USA, pp:335-342

Bassil, R., Hobeica, R., Itani, W., Ghali, C., Kayssi, A., Chehab, A., 2012. Security analysis and solution for thwarting cache poisoning attacks in the Domain Name System, in: 2012 19th International Conference on Telecommunications (ICT). pp. 1–6. https://doi.org/10.1109/IC℡.2012.6221233

Chandramouli, R., Rose, S., 2013. Secure Domain Name System (DNS) Deployment Guide (No. NIST SP 800-81-2). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-81-2

Chandramouli, R., Rose, S., 2006. Challenges in securing the domain name system. IEEE Secur. Priv. 4, 84–87.

Chandramouli, R., Rose, S., 2005. An integrity verification scheme for DNS zone file based on security impact analysis, in: 21st Annual Computer Security Applications Conference (ACSAC’05), p. 10 pp.-321. https://doi.org/10.1109/CSAC.2005.9

Cheung, S., Levitt, K.N., 2000. A formal-specification based approach for protecting the domain name system, in: Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference On. IEEE, pp. 641–651.

Dagon, D., Antonakakis, M., Vixie, P., Jinmei, T., Lee, W., 2008. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries, in: Proceedings of the 15th ACM Conference on Computer and Communications Security. ACM, pp. 211–222.

Gersch, J., Massey, D., 2009. Deploying DNS Security (DNSSEC) in Large-Scale Operational Environments, in: 2009 Cybersecurity Applications Technology Conference for Homeland Security. pp. 169–180. https://doi.org/10.1109/CATCH.2009.43

Gupta, B., Agrawal, D., Yamaguchi, S. (2016) Handbook of research on modern cryptographic solutions for computer and cyber security. IGI Global, 2016, ISBN13: 9781522501053

Gupta, B. (2017) Smart XSS attack surveillance system for OSN in virtualized intelligence network of nodes of fog computing." International Journal of Web Services Research (IJWSR) 14.4 (2017): 1-32.

Gupta, B. (2018) Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives. CRC Press, Taylor & Francis, ISBN: 9780815371335, 2018.

Kakoi, K., Jin, Y., Yamai, N., Kitagawa, N., Tomoishi, M., 2016. Design and Implementation of a Client Based DNSSEC Validation and Alert System, in: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), pp. 8–13. https://doi.org/10.1109/COMPSAC.2016.91

Kalafut, A.J., Shue, C.A., Gupta, M., 2011. Touring DNS Open Houses for Trends and Configurations. IEEEACM Trans. Netw. 19, 1666–1675. https://doi.org/10.1109/TNET.2011.2130537

Krishnaswamy, S., Hardaker, W., Mundy, R., 2009. DNSSEC in Practice: Using DNSSEC-Tools to Deploy DNSSEC, in: 2009 Cybersecurity Applications Technology Conference for Homeland Security, pp. 3–15. https://doi.org/10.1109/CATCH.2009.21

Larsen, M.V., Gont, F., n.d. Port Randomization [WWW Document]. URL https://tools.ietf.org/html/rfc6056 (accessed 10.28.17).

Larson, M., Massey, D., Rose, S., Arends, R., Austein, R., 2005. Resource Records for the DNS Security Extensions [Online]. URL https://tools.ietf.org/html/rfc4034 (accessed 10.19.17).

Lian, W., Rescorla, E., Shacham, H., Savage, S., 2013. Measuring the Practical Impact of DNSSEC Deployment., in: USENIX Security Symposium. pp. 573–588.

Margaret Rouse, 2017. What is promiscuous mode? [Online]. SearchSecurity. URL http://searchsecurity.techtarget.com/definition/promiscuous-mode (accessed 12.2.17).

Memos, V., Psannis, K., Ishibashi, Y., Kim, B,, Gupta, B. (2017) An efficient algorithm for media-based surveillance system (EAMSuS) in IoT smart city framework. Future Generation Computer Systems, Vol. 83, pp: 619-628.

Nation, J. (2018) Major DNS Spoofing Hack Affects Amazon Web Services. Metacert. [Online] Available at <https://medium.com/metacert/major-dns-spoofing-hack-affects-amazon-web-services-157e3565c844> (Date Accessed: 26-3-19)

Pa, Y., Yoshioka, K., Matsumoto, T., 2013. Search Engine Based Investigation on Misconfiguration of Zone Transfer, in: 2013 Eighth Asia Joint Conference on Information Security, pp. 56–62. https://doi.org/10.1109/ASIAJCIS.2013.16

Plageras, A., Psannis, K., Stergiou, C., Wang, H., Gupta, B. (2018) Efficient IoT-based sensor BIG Data collection–processing and analysis in smart buildings. Future Generation Computer Systems, 82, 349-357, 2018 https://doi.org/10.1016/j.future.2017.09.082

Hubert, A., van Mook, R. (2009) Measures for Making DNS More Resilient against Forged Answers. RFC 5452, 2009 [Online] Available at <https://tools.ietf.org/html/rfc5452> (Date Accessed: 5-4-2019)

Rose, S., Nakassis, A., 2008. Minimizing information leakage in the DNS. IEEE Netw. 22, 22–25. https://doi.org/10.1109/MNET.2008.4476067

Rostampour, S., 2012. Deploying DNS Security Extensions. Thesis. Online. Available at: http://studentarbeten.chalmers.se/publication/173693-deploying-dns-security-extensions

Rouse, M., 2015. Software Defined Networking [WWW Document]. URL https://searchsdn.techtarget.com/definition/software-defined-networking-SDN (accessed 4.30.18).

Satam, P., Alipour, H., Al-Nashif, Y., Hariri, S., 2015. DNS-IDS: Securing DNS in the Cloud Era, in: 2015 International Conference on Cloud and Autonomic Computing, pp. 296–301. https://doi.org/10.1109/ICCAC.2015.46

Shulman, H., Waidner, M., 2014. Towards Forensic Analysis of Attacks with DNSSEC, in: 2014 IEEE Security and Privacy Workshops. Presented at the 2014 IEEE Security and Privacy Workshops, pp. 69–76. https://doi.org/10.1109/SPW.2014.20

Stergiou, C., Psannis, K., Kim, B. & Gupta, B. (2016). Secure Integration of Internet-of-Things and Cloud Computing. Future Generation Computer Systems, Vol. 4, No. 2, pp: 964-975

Tewari, A., Gupta, B. (2017) Cryptanalysis of a novel ultra-lightweight mutual authentication protocol for IoT devices using RFID tags. The Journal of Supercomputing 73, no. 3, pp: 1085-1102.

Ubuntu, 2010. Ubuntu Manpage: Haveged [WWW Document]. URL http://manpages.ubuntu.com/manpages/xenial/man8/haveged.8.html (accessed 2.5.18).

Wijngaards, W.C.A., Overeinder, B.J., 2009. Securing DNS: Extending DNS Servers with a DNSSEC Validator. IEEE Secur. Priv. 7, 36–43. https://doi.org/10.1109/MSP.2009.133

Yu, X., Chen, X., Xu, F., 2011. Recovering and Protecting against DNS Cache Poisoning Attacks, in: 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, pp. 120–123. https://doi.org/10.1109/ICM.2011.266

Zou, F., Zhang, S., Pei, B., Pan, L., Li, L., Li, J., 2016. Survey on Domain Name System Security, in: 2016 IEEE First International Conference on Data Science in Cyberspace (DSC), pp. 602–607. https://doi.org/10.1109/DSC.2016.96