

DNS CLIENT WITH AUTHENTICATION SERVER

A PROJECT REPORT

Submitted by

Sayak Sengupta 19BCI0270

Deepanshu Saini 19BCI0248

Sridhar Reddy 19BCI0272

Course Code: CSE 2008

Course Title: Network Security

Under the guidance of

Dr. S. Anto

Associate Professor, SCOPE,

VIT , Vellore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

JUNE, 2021

INDEX

Table of Contents	Page no
Abstract	5
1. Introduction 1.1 Overview 1.2 Objective 1.3 Applications	6
2. Literature Survey 2.1 Problem Definition 2.2 Table of Contents	7-9
3. Overview of the Work 3.1 Objectives of the Project 3.2 Software Requirements 3.3 Hardware Requirements	10
4. System Design 4.1 Algorithms, Block Diagrams etc 4.2 Proposed Algorithm	11-12
5. Implementation 5.1 Description of Modules/Programs 5.2 Source Code 5.3 Test Cases	13-19
6. Output and Performance Analysis 6.1 Execution in Snapshots 6.2 Output - in terms of performance metrics	20

7. Conclusions and Future Decisions	21
8. References	22

Abstract

DNS is a TCP/IP protocol used on different platforms. The domain name space is divided into three different sections: generic domains, country domains, and inverse domain. DNS cache poisoning, also known as DNS spoofing, is a type of attack that exploits vulnerabilities in the domain name system (DNS) to divert Internet traffic away from legitimate servers and towards fake ones. DNS cache poisoning is the act of entering false information into a DNS cache, so that DNS queries return an incorrect response and users are directed to the wrong websites.

Assuming an entity needs to spoof the identity of some. Other entities, it's enough to vary the mapping between its low level address and its high level name. It means that an assailant will pretend the name of somebody by modifying the association of his address from his own name to the name he needs to impersonate. Once an assailant has done that, an appraiser will no longer distinguish between truth and fake entities ex->google.com and gogle.com. DNS is a TCP/IP protocol used on different platforms. The domain name space is divided into three different sections: generic domains, country domains, and inverse domain.

1. Introduction:

1.1. Overview:

This project aims at creating an authentication system which will detect any DNS spoofing or DNS cache poisoning and prevent the server from returning an incorrect result record or IP Address. This would thereby provide a secure server for the host to surf on the internet without any hassle related to any kind of DNS attacks, hence making the system robust.

1.2. Objectives of the Project:

The main objective of the project is to provide a DNS attack free environment to the user such that he/she can surf on the internet without having to face attacks like DNS cache poisoning or DNS spoofing by providing the user with the authentication server which is simple to implement and thereby would provide the IP address of verified websites and block unwanted or malicious IP address requests.

1.3. Application of the Project:

- 1.3.1. The DNS Client is capable of resolving the IP address of a host from the host's name. It does this by sending DNS requests to a DNS Server. The IP address of a DNS Server is specified in the network interface configuration file or can be obtained from the DHCP Server for the Local Area Network.
- 1.3.2. The DNS Client is capable of resolving the IP address of a host from the host's name. It does this by sending DNS requests to a DNS Server. The IP address of a DNS Server is specified in the network interface configuration file or can be obtained from the DHCP Server for the Local Area Network.
- 1.3.3. The DNS Client caches the resolved IP addresses. The length of time the resolved host IP address is kept in the local cache depends on the Time to Live (TTL) timeout. This is returned in an answering packet from the DNS Server. The next time a DNS is requested, the cache table is checked first. If a valid host is found, the IP address is resolved from the cache and no actual DNS request is sent to the DNS Server.

2. Literature Survey:

2.1. Problem Definition:

Although, a big vulnerability was found in the Windows DNS Server protocol. Its concept is very similar to SMBGHOST (which you can chain to SMBleed using our technical walkthrough). This vulnerability can lead to Remote Code Execution, which raises its impact to critical levels and also its exploitation complexity is somewhere at easy to medium.

Using the Python script from Github, we can host our own malicious domain, and after that, it will work as following:

- From the attacker's machine, send a SIG-records DNS query for the malicious domain through the vulnerable DNS server using this command: `nslookup -type=sig {subdomain.attacker_domain} {vulnerable server IP}`
- The vulnerable DNS server will ask a public DNS server about the nameservers for your malicious domain
- The nameservers for the attacker's DNS server are returned to the vulnerable Windows DNS Server
- The vulnerable DNS will act as a client and forward the initial SIG request query to the Attacker's DNS Server
- Through our script, the malicious DNS Server will respond with a malicious SIG value, crafted to produce a buffer overflow and crash the DNS service on the victim server.

2.2. Table of Contents:

Reference	Problems	Solution Proposed
<p>DNS Security: Antonio Lioy, Fabio Maino, Marius Marian, Daniele Mazzocchi, Dipartimento di Automatica e Informatica Politecnico di Torino Torino (Italy)</p> <p>(Research Paper)</p>	<p>DNS security</p>	<p>The present research paper is an overview of the security problems affecting the Domain Name System and also of the solutions developed throughout the last years in order to provide a better, trustworthy and safer name resolution protocol for the expanding Internet community of present days. The paper discusses the basic notions regarding DNS and introduces the reader to the known security threats regarding DNS. The DNSSEC subset proposed is presented and analyzed from both theoretical and practical points of view, explaining the existing security features of the implementations available today.</p>
<p>Security System for DNS using Cryptography : Sachin Kumar Sinha, Avinash Kant Singh, Amaresh Sharma B.TECH(IT), ITM Gida Gorakhpur, GBTU</p> <p>(Research Paper)</p>	<p>Secure system for DNS client</p>	<p>The mapping or binding of IP addresses to host names became a major problem in the rapidly growing Internet and the higher level binding effort went through different stages of development up to the currently used Domain Name System (DNS).The DNS Security is designed to provide security by combining the concept of both the Digital Signature and Asymmetric key (Public key) Cryptography. Here the Public key is sent instead of the Private key. The DNS security uses Message Digest Algorithm to compress the Message(text file) and PRNG(Pseudo Random Number Generator) Algorithm for</p>

		<p>generating Public and Private key. The message combines with the Private key to form a Signature using DSA Algorithm, which is sent along with the Public key. The receiver uses the Public key and DSA Algorithm to form a Signature. If this Signature matches with the Signature of the message received, the message is Decrypted and read else discarded.</p>
--	--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3. Overview of the Work:

3.1. Objectives of the project:

To create an authentication system which will detect any DNS spoofing or DNS cache poisoning and prevent the server from returning an incorrect result record or IP Address. This would in return give us a secured network which would allow the user to surf on the internet without the hassle of any DNS attacks by hackers. Also, the idea is to incorporate a DNS-free environment on the internet which would be a breakthrough in the field of DNS security.

3.2. Software Requirements:

The Authentication Server would be using the socket header file in Python to create the server which would request for packets(IP addresses) from websites thereby detecting whether or not any DNS attack is taking place or not.

Softwares used:

Visual Studio Code (VS Code)

Programming language used:

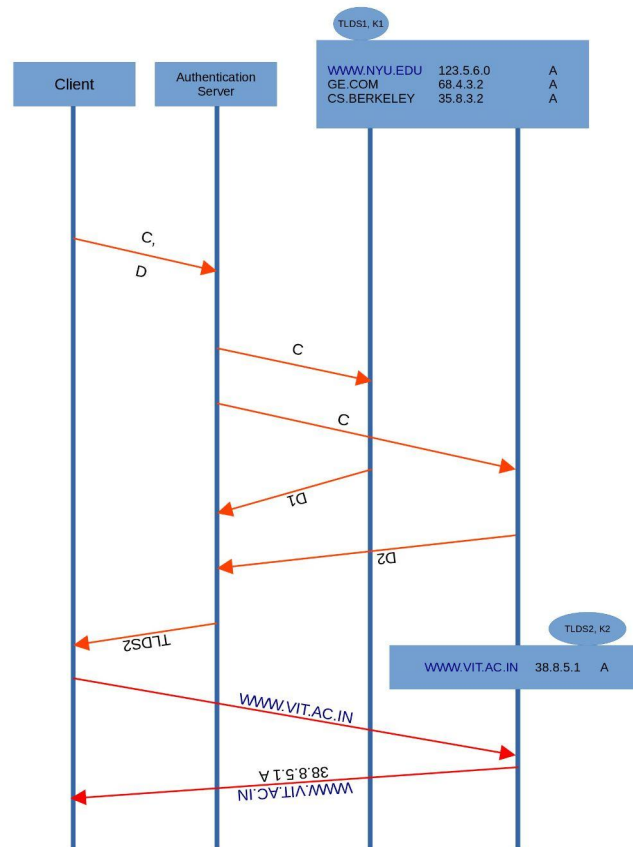
Python

3.3. Hardware Requirements:

The hardware interface for the user would be any PC having a configuration of P-IV and above 2GB HDD for loading any OS so that it can interact with the system without any problem. The main interface would be monitor, keyboard and mouse.

4. System Design

4.1. Algorithms and Block Diagrams:



4.2. Proposed Algorithm:

The DNS client program uses a key and a challenge string to create a digest and sends the challenge string as well as the digest to the authentication server. The authentication server sends ONLY the challenge string to both the TLDS servers and the TLDS servers return the respective digests. The TLDS servers each store a key and only one of them matches the key that the client used to create the digest. Hence, only one of the responses from the TLDS servers

match the digest sent by the client. The Authentication server (root server) sends the hostname of the TLDS server with the correct match to the client. The DNS client then connects to that TLDS server with a query string and obtains the A record (if found) from the authenticated TLDS server. The two TS servers each maintain a DNS_table consisting of three fields: Hostname, IP address, Flag (A). In addition, the TS servers each maintain a key which is used to create a digest from a challenge. When the DNS client connects to the authenticated TLDS server, it sends the hostname as a string. The TLDS server does a look up in the DNS_table and if there is a match, sends the DNS table entry as a string ["Hostname IPaddress A"]. If the host name does not exist, then an error string Error: HOST NOT FOUND is returned. Note, that in this Project, the DNS client connects to the TLDS server to get the IP address for a given hostname. The hostname string, the Key and the challenge will be given one per line in a file (PROJ3-HNS.txt) and the keys one per line will be in a file PROJ3-KEY1.txt and PROJ3-KEY2.txt. The DNS table entries will also be one per line and will be in PROJ3-TLDS1.txt and PROJ3-TLDS2.txt. The TLDS servers, in addition to reading the DNS entries, will each obtain its key by reading the value from the corresponding key files. TLDS1 will read the key from PROJ3-KEY1.txt and TLDS2 will read the key from PROJ3-KEY2.txt. Your client program should output the results to a file RESOLVED.txt. As part of your submission, you need to submit four program files as well as the output file. The DNS client will read from PROJ3-HNS.txt. Each line will contain a key, a challenge and a hostname (k3521 arianna www.princeton.edu). Using the key and the challenge, the DNS client creates a digest. Then connects to the AS server and sends ONLY the challenge string and the digest to the AS server. The AS server then sends ONLY the challenge string to both the TLDS servers and collects the responses (digests). Then it does a digest comparison and sends the name of the TLDS server that returns the correct digest. Note: as the keys read by the TLDS servers are different, only one digest will match. The DNS client then sends the hostname part of the query string to that TLDS server returned by the AS server and writes the response received from the TLDS server and the hostname of the TLDS server to the output file RESOLVED.txt.

5. Implementation:

5.1. **Description of Module/Program:**

We aim create a program so as to provide a DNS attack free environment to the user such that he/she can surf on the internet without having to face attacks like DNS cache poisoning or DNS spoofing by providing the user with the authentication server which is simple to implement and thereby would provide the IP address of verified websites and block unwanted or malicious IP address requests. The program would initiate the server from the client sidesend back the website addresses along with their authentic IP addresses and returning NOT FOUND message for those websites which either do not exist or those which are means of any DNS attack to the user and we can differentiate based on A for authenticated and NA for not-authenticated.

5.2. **Source Code:**

Server.py:

```
import socket as mysoc
import pickle
import sys
import hmac

tlds1 = "TLDS1"
tlds2 = "TLDS2"

def rs():
    try:
        asssd = mysoc.socket(mysoc.AF_INET, mysoc.SOCK_STREAM)
    except mysoc.error as err:
        print('[RS] {} \n'.format("RS server socket open error ", err))
    try:
        astotscom = mysoc.socket(mysoc.AF_INET, mysoc.SOCK_STREAM)
    except mysoc.error as err:
        print('{} \n'.format("socket open error ", err))
```

```

try:
    astotsedu = mysoc.socket(mysoc.AF_INET, mysoc.SOCK_STREAM)
except mysoc.error as err:
    print('{} \n'.format("socket open error ", err))

AS_table = {}
host = mysoc.gethostname()
com = tlds1
AS_table[com] = {'ip': mysoc.gethostbyname(host), 'flag': 'NS'}
edu = tlds2
AS_table[edu] = {'ip': mysoc.gethostbyname(host), 'flag': 'NS'}

try:
    sa_sameas_myaddr = AS_table[com]['ip']
    portTLDS1 = 1270
    server_binding = (sa_sameas_myaddr, portTLDS1)
    #astotscom.connect(server_binding)
except mysoc.error as err:
    print('{} \n'.format("TLDS1 connect error "), err)
    exit()

try:
    sa_sameas_myaddr = AS_table[edu]['ip']
    portTLDS2 = 5677
    server_binding = (sa_sameas_myaddr, portTLDS2)
    # astotsedu.connect(server_binding)
except mysoc.error as err:
    print('{} \n'.format("TLDS2 connect error "), err)
    exit()

server_binding = ("", 50008)
assd.bind(server_binding)
assd.listen(1)
host = mysoc.gethostname()

```

```

print("[S]: Authentication Server host name is: ", host)
localhost_ip = (mysoc.gethostbyname(host))
    print("[S]: Attempting to connect to client.\n[S]: Server IP address is
",localhost_ip)
    casd, addr = assd.accept()
    print("[S]: Got a connection request from a client at", addr)

server = ""
# Authentication Loop

while True:
    data = casd.recv(100)
    if not data: break
    digest = pickle.loads(data)
    key = digest[0]
    challenge = digest[1]
    dvalid = hmac.new(key.encode(), challenge.encode("utf-8"))
    astotscom.send(pickle.dumps(challenge))
    digestTLDS1 = pickle.loads(astotscom.recv(100))
    astotsedu.send(pickle.dumps(challenge))
    digestTLDS2 = pickle.loads(astotsedu.recv(100))
    if dvalid.hexdigest() == digestTLDS1:
        server = "TLDS1"
        astotscom.send(pickle.dumps("pass"))
        astotsedu.send(pickle.dumps("fail"))
    elif dvalid.hexdigest() == digestTLDS2:
        server = "TLDS2"
        astotscom.send(pickle.dumps("fail"))
        astotsedu.send(pickle.dumps("pass"))
    else:
        server = "error"
        astotscom.send(pickle.dumps("fail"))
        astotsedu.send(pickle.dumps("fail"))
    casd.send(pickle.dumps(server))

```

```
# close everything
astotscom.close()
astotsedu.close()
assd.close()
```

```
rs()
```

Client.py

```
import socket as mysoc
import pickle
import sys
import time
```

```
def client():
```

```
    try:
```

```
        ctots = mysoc.socket(mysoc.AF_INET, mysoc.SOCK_STREAM)
```

```
    except mysoc.error as err:
```

```
        print('{} \n'.format("socket open error ", err))
```

```
    # [tlds1 socket]
```

```
    try:
```

```
        ctots1 = mysoc.socket(mysoc.AF_INET, mysoc.SOCK_STREAM)
```

```
    except mysoc.error as err:
```

```
        print('{} \n'.format("socket open error ", err))
```

```
    # [tlds2 socket]
```

```
    try:
```

```
        ctots2 = mysoc.socket(mysoc.AF_INET, mysoc.SOCK_STREAM)
```

```
    except mysoc.error as err:
```

```
        print('{} \n'.format("socket open error ", err))
```

```
    try:
```

```
        file_name = "SAYAK-HNS.txt"
```



```

    fr = open(file_name, "r")
except IOError as err:
    print('{} \n'.format("File Open Error ", err))
    print("Please ensure table file exists in source folder")
    exit()

host = mysoc.gethostname()
sa_sameas_myaddr = mysoc.gethostbyname(host)
portAS = 50008
portTLDS1 = 5679
portTLDS2 = 5676

# [bind ctors socket to RS address, RS port]
try:
    server_binding = (sa_sameas_myaddr, portAS)
    #ctoas.connect(server_binding)
except mysoc.error as err:
    print('{} \n'.format("connect error "), err)
    exit()
first1 = True

if first1:
    first1 = False
    try:
        server_binding = (sa_sameas_myaddr, portTLDS1)
        #ctots1.connect(server_binding)
    except mysoc.error as err:
        print('{} \n'.format("connect error "), err)
        exit()

first2 = True
if first2:
    first2 = False
    try:

```

```

server_binding = (sa_sameas_myaddr, portTLDS2)
#ctots2.connect(server_binding)
except mysoc.error as err:
    print('{} \n'.format("connect error "), err)
    exit()
with open("RESOLVED.txt", "w") as fw:
    for hostname in fr:
        digest = hostname.split()
        ctoas.send(pickle.dumps(digest))
        dataFromAS = pickle.loads(ctoas.recv(100))
        if "TLDS1" in dataFromAS:
            ctots1.send(pickle.dumps(digest[2]))
            data = pickle.loads(ctots1.recv(100))
            fw.write(data + '\n')
        elif "TLDS2" in dataFromAS:
            ctots2.send(pickle.dumps(digest[2]))
            data = pickle.loads(ctots2.recv(100))
            fw.write(data + '\n')
        else:
            fw.write(digest[2] + " - No servers matching key\n")
# close everything
time.sleep(5)
fr.close()
fw.close()
ctots1.close()
ctots2.close()
ctoas.close()
exit()

client()

```

5.3. Test Cases

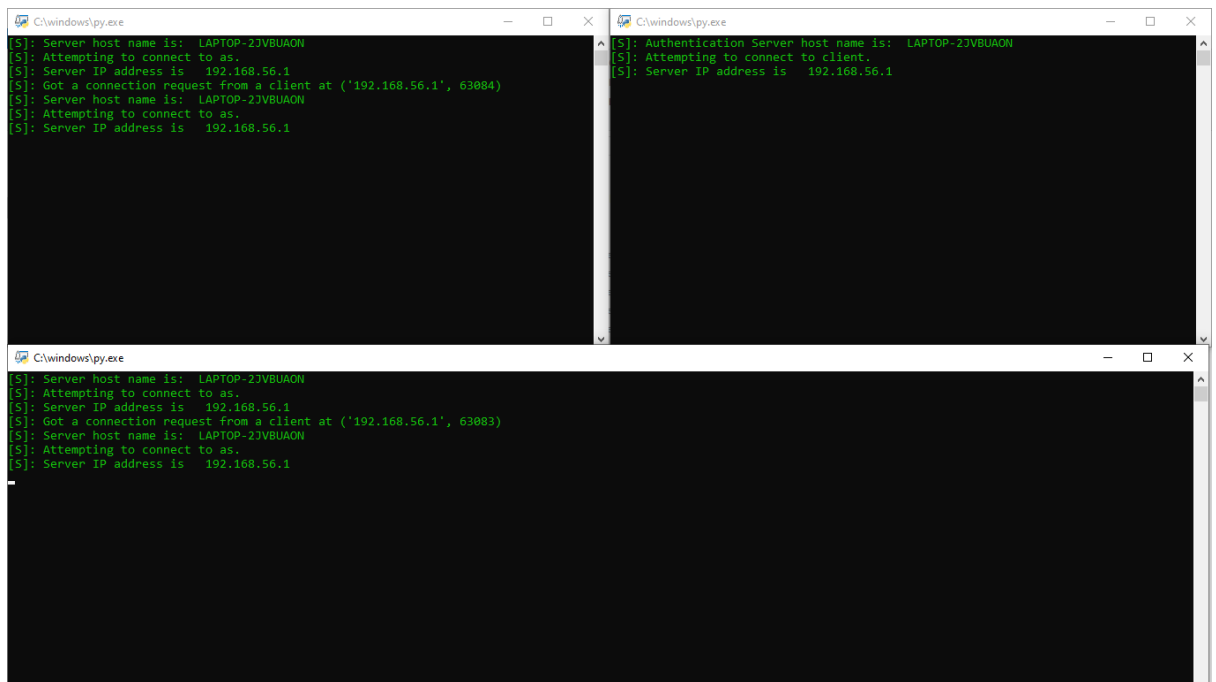
We would require input of some default websites along with their IP addresses as reference points such that when the server requests for the IP addresses from these websites through packets, then in return it would verify whether the IP addresses sent by them are verified or not and hence would display it to the user with an authorized server denoted by A.

Test cases	Expected results	Authentication(A)
www.facebook.com	192.64.4.2	A
www.mit.edu	8.7.45.2	A
www.google.com	8.6.4.2	A
www.yotbe.com	Error:HOST NOT FOUND	NA

6. Output and Performance Analysis:

6.1. Execution:

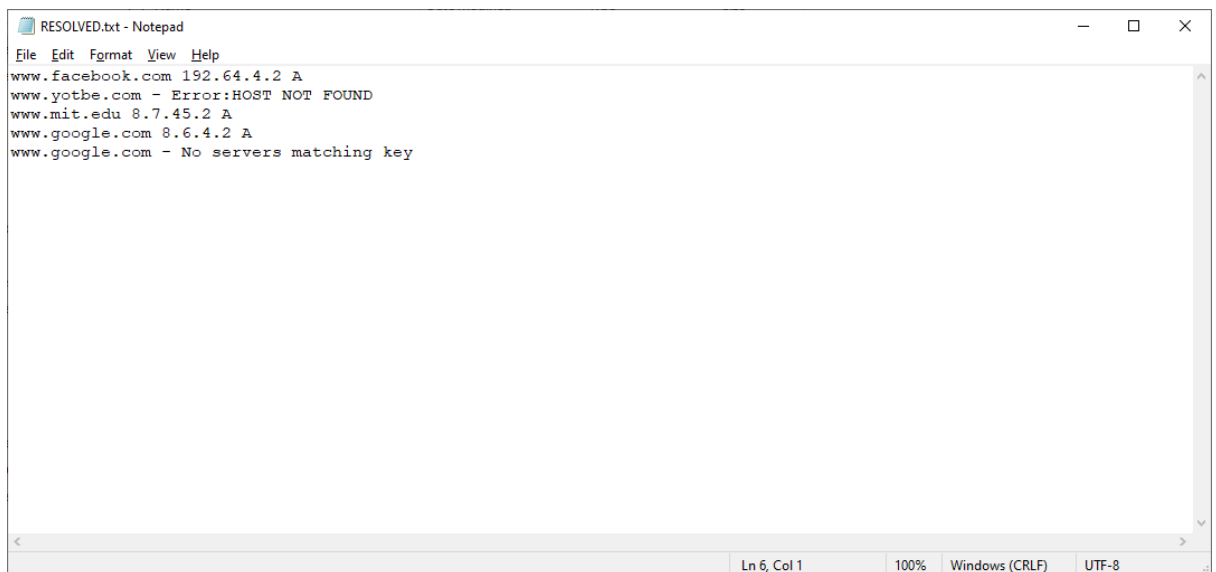
Running TLDS1 and TLDS2 Servers :



```
C:\windows\py.exe
[5]: Server host name is: LAPTOP-2JVBUAON
[5]: Attempting to connect to as.
[5]: Server IP address is 192.168.56.1
[5]: Got a connection request from a client at ('192.168.56.1', 63084)
[5]: Server host name is: LAPTOP-2JVBUAON
[5]: Attempting to connect to as.
[5]: Server IP address is 192.168.56.1

C:\windows\py.exe
[5]: Authentication Server host name is: LAPTOP-2JVBUAON
[5]: Attempting to connect to client.
[5]: Server IP address is 192.168.56.1

C:\windows\py.exe
[5]: Server host name is: LAPTOP-2JVBUAON
[5]: Attempting to connect to as.
[5]: Server IP address is 192.168.56.1
[5]: Got a connection request from a client at ('192.168.56.1', 63083)
[5]: Server host name is: LAPTOP-2JVBUAON
[5]: Attempting to connect to as.
[5]: Server IP address is 192.168.56.1
```



```
RESOLVED.txt - Notepad
File Edit Format View Help
www.facebook.com 192.64.4.2 A
www.yotbe.com - Error:HOST NOT FOUND
www.mit.edu 8.7.45.2 A
www.google.com 8.6.4.2 A
www.google.com - No servers matching key

Ln 6, Col 1 100% Windows (CRLF) UTF-8
```

6.2. Output (in terms of performance metrics):

Time taken to authenticate www.facebook.com : 1.3s

Time taken to authenticate www.mit.edu : 1.45s

Time taken to authenticate www.google.com : 1.14s

Average time taken to authenticate 10 websites: 1.27s per website

7. **Conclusion:**

We were able to successfully create a program that allows the client to call an authentication server for dns spoofing prevention. It also tells you which site has no host name or isn't authenticated.

We believe DNS would make a good distribution point of application keys and certificates for large scale systems. The main reason is that DNS is a unique provider of bindings between commonly used names. We presented a proposal for DNSSEC that, when properly implemented, offers the highest level of security while reducing network traffic. In addition, it reduces storage requirements and enables efficient mutual authentication. In particular, for highly critical parts of the DNS, like root servers or other servers near the root, our service can provide increased security. We rely on DNSSEC to provide authenticated delegation, while keeping the functional overhead of key distribution outside the critical DNS infrastructure. This strategy allows us to use the name service infrastructure to guarantee authenticity.

8. **References:**

[1] Sachin Kumar Sinha, Avinash Kant Singh,..., “**Security System for DNS using Cryptography**”, International Journal of Engineering Research & Technology (IJERT) ,ISSN: 2278-0181 , February, 2013

[2]Antonio Lioy, Fabio Maino,..., “**DNS Security**”, International Journal of Engineering Research & Technology (IJERT), Publication details: Terena Networking Conference, May 22-25, 2000, July, 2001

[3]Book referred : Cryptography and Network Security: Principles and Practice