DCG: A Client-side Protection Method for DNS Cache

Yan Zhao, Ning Hu*, Chi Zhang, and Xinda Cheng
Guangzhou University, Guangzhou, 510006 China
{2111906107, huning, 2111906100, 2111906003}@e.gzhu.edu.cn

Abstract

Domain name system provides resolution services between domain names and IP addresses for internet applications and it is the backbone of the modern internet. Since the security of domain name system is critical to the internet, a large number of solutions have emerged. Unfortunately, most of these works are focused on server-side protection, but few solutions for client protection. Because the server-side solution cannot guarantee that the client uses a trusted domain name, this paper proposes a client-side protection method for domain name system cache. Our solution monitors the local cache of domain name system in real time and asynchronously verifies the authenticity of each name resolution result through a trusted third party. Experimental results show that our method can resist domain name poisoning attacks against clients. And our solution is fully compatible with the existing domain name system, and has good incremental deployment capabilities.

Keywords: DNS Security, DNS spoofing, Cache Verification

1 Introduction

Domain Name System (DNS) is an important basic service of the Internet [13]. The main function of DNS is to provide the mapping of domain name and IP address. Almost all Internet applications use DNS. Everyone relies on the services provided by DNS when they go online. DNS is so widely used, any attack on DNS will have serious consequences. Therefore, DNS security is the foundation of Internet security [14]. However, at the beginning of the design of DNS, it was assumed that it would run in a trusted network environment, and nobody considered security issues. As a result, the events of using DNS vulnerabilities to conduct network attacks are endless. This is a serious threat to the security of the Internet [2].

Since the researchers did not consider DNS security at the beginning of the design, many attacks against DNS appeared. Such as DNS spoofing [10], DNS hijacking [3], and DNS cache poisoning [15]. In order to improve the security of DNS, many methods have been proposed, such as DNSSEC [17], HTTPS-over DNS [6], TLS-over DNS [7], and DNScrypt [4]. However, these methods have some limitations, such as focusing only on the security of the DNS server and failing to protect the DNS client, incompatible with the existing DNS, and difficult to promote. DNSSEC has not been extended to the client and cannot protect the security of the DNS client. HTTPS-over DNS , TLS-over DNS , and DNScrypt modify the DNS protocol, making it difficult to promote and deploy. And these technologies encrypt DNS messages, making it difficult for the organization's network security department to monitor malicious DNS traffic. With the development of blockchain, some researchers have proposed DNS security solutions based on blockchain [11]. But this direction is still in the research stage.

Alharbi et al. [1] proposed a cache poisoning attack against DNS clients, which can bypass existing DNS defense solutions. That the neglect of DNS security of client is the reason for the success of the

Journal of Internet Services and Information Security (JISIS), volume: 10, number: 2 (May 2020), pp. 103-121
DOI: *10.22667/JISIS.2020.05.31.103*

*Corresponding author: Cyberspace Institute of Advanced Technology, Guangzhou University, 230 Wai Huan Xi Road, Guangzhou Higher Education Mega Center, Guangzhou, 510006, China, email: huning@e.gzhu.edu.cn.

attack. Therefore, we need a new way to deal with such attacks. An easy way is to verify the DNS cache. Every time the user accesses a domain name, the domain name and its IP address will be cached by the computer. Therefore, as long as we verify that the client-side DNS cache is correct, we can ensure that the IP address of the domain name is correct. So, the key problem is how to verify the correctness of the client's DNS cache. If it can be verified, almost all DNS spoofing attacks can be closed.

In this paper, we propose DCG, which is the abbreviation of DNS cache guardian. And DCG as a supplement to existing DNS defense strategies can protect DNS on the client-side. The core idea of this method is to verify the DNS cache. When a new record enters the cache, the client sends the record to a third-party server through a secure channel. The third-party server will verify the record and return the verification result to the client. In this way, the client has the ability to distinguish fake DNS records. Note that after receiving the DNS reply, the user will use the IP address to access the website first, and then the user will receive the DCG verification result. This is a compromise solution. The purpose is to provide users with security services as much as possible without sacrificing user experience. When a user receives a fake IP address for a domain name, the most likely possibility is that the IP address is a phishing website. The hacker wants the user to enter the account password on his phishing website and steal user's property. But it takes time for users to enter information. If the DCG can complete the verification process in a short time, the user can be warned before the user completes the input. Our experiment shows that DCG only needs about 1s to produce results. User input information usually takes more than 1s.

The idea of DCG is terminal defense. Even if the DNS security mechanism in the DNS server or LAN is bypassed by hackers, this method can still protect DNS in the last step. In addition, this method is compatible with DNS and can work normally in the environment of Content Delivery Network. The contributions of this paper are as follows:

- Introduce DNS risks and existing security mechanisms, and analyze the limitations of those mechanisms.
- Propose a method to protect DNS at client-side

The first part of this paper is the introduction. The main content of this paper is briefly introduced in this part. The second part is the motivation. This part will briefly introduce DNS, analyze the causes of DNS insecurity and the limitations of existing defense strategies, and finally give the goal of our method. In the third part, the DCG is described in detail. In the fourth part, we show some experiments and experimental results. Part 5 is related work, some work similar to this paper will be introduced. Part 6 will elaborate on issues that have not been discussed carefully or that have not been resolved temporarily. The last section is the conclusion.

2 Motivation

2.1 DNS Overview

Domain Name System (DNS) is an important basic service of the Internet. DNS protocol is based on the UDP / TCP protocol. As a large distributed database on the Internet, the main function of DNS is to map the domain name and IP address. Each IP address can have a host domain name. The host domain name consists of one or more strings, separated by a decimal point. With the host domain name, as long as you remember the relatively intuitive and meaningful host name, users can query the domain name through DNS, and then DNS will return the information including the corresponding IP address to users. DNS will take users away from the boring and difficult to remember IP address and make it easier for

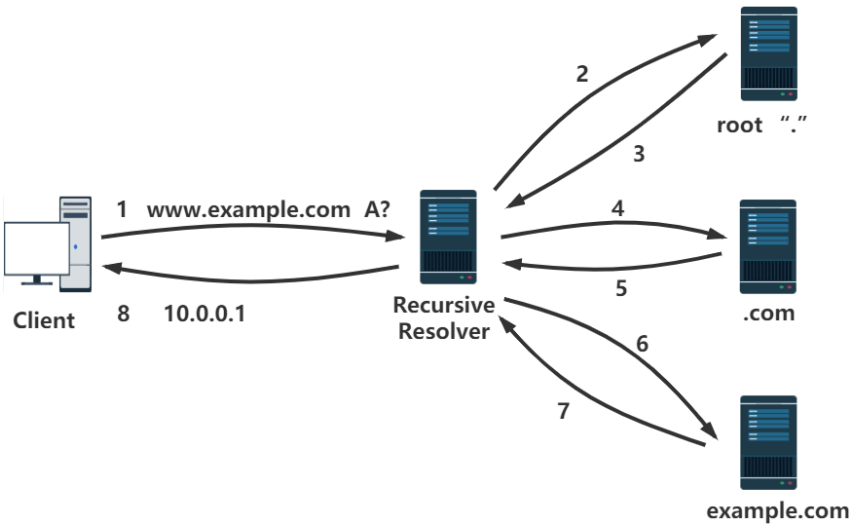

Figure 1: The process of DNS

users to access the Internet. The working process of DNS is shown in Figure 1. When the client accesses a domain name (www.example.com), the client will first search the DNS cache (browser cache, system cache and hosts file). If it is not found in cache, client will send a request to the recursive resolver. Then, the recursive resolver will first look for the cache, and if there is, resolver will reply to the client. If there is not , the recursive resolver will send the request to the root server, top-level domain name server, authoritative domain name server in turn, until the example.com domain name server is found. Finally, the recursive resolver sends the result(10.0.0.1) to the client.

DNS is a public service system. The services it provides are open and transparent to all users on the Internet and can be accessed and used by anyone. Therefore, its original designers did not consider its security, and all designs were designed with convenience. However, with the development of Internet, the scale of the Internet has been expanding, and the value of information on the Internet has become larger and larger, attracting many criminals to attack it. Today, a variety of DNS security vulnerabilities have seriously threatened the privacy and interests of Internet users. As the basic service of the Internet, the security of DNS is directly related to whether the Internet system can operate normally.

DNS message is transmitted in plain text, without encrypting and signing, and is extremely vulnerable to spoofing attacks. For example, attackers can make most of the network applications paralyzed by sending many fake DNS responds. This is known as DNS spoofing. Currently, DNS spoofing attacks are mainly in the form of internal attacks and serial number attacks. The internal attack means that the attacker successfully invades and controls the DNS server. The domain name in the database can be modified by the attacker at will, and the IP address corresponding to the domain name is converted into the IP address preset by the attacker. When the host user queries the domain name, what user get is the fake IP preset by the attacker. The serial number attack is based on a vulnerability in the DNS server. In the DNS protocol, domain name request and response packets are matched by serial number. The attacker masquerades as a DNS server and passes the response packet with the request packet serial number to the host user, then forwards it to the host client before the real response packet arrives. After that, the IP address corresponding to the relevant domain name queried by the host user turns to be the fake address specified by the attacker, and the user is naturally taken to the website preset by the attacker.

The targets of DNS spoofing attacks are mostly clients, while the targets of DNS cache poisoning attacks are DNS recursive servers. Therefore, DNS cache poisoning attacks will cause more serious

consequences. At present, the DNS uses the UDP protocol to transmit query and response data packets. It uses a simple trust mechanism to acknowledge the IP address, port, and random query ID of the original query packet for the first received response packet, without doing any analysis on the legality of the packet. If the packet matches, DNS accepts it as the correct response packet, and continues the DNS resolution process, then discards all subsequent response packets. This allows the attacker to fake the authoritative name server and send a fake reply packet to the cached DNS server, and strive to complete the response to contaminate the DNS cache. If the forged response packet sent by the attacker reaches the cached DNS server before the correct response packet sent by the authoritative name server, and matches the original query packet IP address, port and random query ID, the DNS cache can be successfully polluted. If the cache of a recursive server is contaminated, all clients using the recursive server are at risk.

2.2 Reasons for DNS insecurity and existing defense mechanism

There are many reasons why DNS security is so fragile. In addition to the problem of the DNS design, there are also many potential security threats caused by human negligence. The main reasons for DNS insecurity can be attributed to the following three aspects:

- **Trust model:** The correct basis for the DNS resolution process is that all DNS servers in the recursive query process provide the correct response and are not being attacked and modified during the data transfer process. This is a hypothetical premise that the DNS protocol is unreasonable. In fact, this trust model is very fragile because the received request may not be sent by the IP address identified in the packet, but an attacker is guessing the Transaction ID by sending a large number of DNS request packets. The response packet may also not be sent by the requested DNS server, but a DNS server response packet forged by a man-in-the-middle attack, or the received request packet has been illegally tampered with. All of these problems are due to the fact that the DNS communication process does not perform any authentication or data integrity verification. This is the root cause of all DNS security problems, and it is also the starting point and foothold for solving them.

- **Simple identification session mode:** The DNS protocol uses the Transaction ID and the source port to identify a session, especially the Transaction ID. This way of identifying sessions can improve communication efficiency, but it is also more vulnerable to attacks. Prior to July 2008, the source port for server communication was not randomly generated. In the old BIND version, the Transaction ID of the pseudo-random sequence can be accurately predicted by observing the Transaction ID in the latest DNS packets. Even in BIND 9, the random algorithm that generates Transaction IDs has vulnerabilities. By observing and analyzing the Transaction IDs in the last 5,000 packets, the probability of success in predicting the next one can be increased to 20%. Therefore, the success rate of DNS attacks by guessing these two fields is still relatively high [1]. The DNS protocol, which relies on two fields for session identification, directly leads to the vulnerability of it.

- **"First Answer Win" principle:** The DNS always believes in the first packet that meets the requirements. That is to say, as long as the above requirements are met and it is the first arriving packet, the DNS server will consider the packet as the correct response packet, and other subsequent response packets will be simply discarded. This is the "First Answer Wins" principle. This principle is the basis for almost all attacks such as DNS spoofing and DNS cache poisoning.

In order to improve the security of DNS, many solutions have been proposed. Domain name system security extensions (DNSSEC) are a series of DNS Security authentication mechanisms provided by

IETF. It provides an extension of source identification and data integrity. DNSSEC relies on the signature to ensure the authenticity and integrity of the DNS response message. The authoritative domain name server signs the resource record with its private key, and the resolution server verifies the received response information with the public key of the authoritative server. If the verification fails, it indicates that the message may be fake, or has been tampered with during the transmission and caching process [9]. However, many years have passed since IETF published the first RFC document about DNSSEC, but the implementation and deployment of DNSSEC are not satisfactory. More importantly, DNSSEC does not extend to the client. It only protects the communication between the recursive resolver and the DNS server, but the communication between the client and the recursive resolver is not protected. Recent defenses such as DNS over HTTPS, DNS over TLS, and DNScrypt, have been proposed primarily to preserve the privacy of DNS traffic. These proposals can also have the side effect of hardening DNS traffic against injection attacks. Although there are standardization efforts behind these proposals, they are not yet widely deployed [1]. DNS is a centralized management system based on the root domain name server. Once the root domain server has problems, the whole Internet will be seriously affected. With the rapid development of blockchain technology, relevant institutions are using the decentralization of blockchain to promote the integration of blockchain technology and DNS [11], but the progress is slow. Existing DNS defense strategies are complex and difficult to promote and deploy. Many defense measures are concentrated on the DNS server, ignoring the terminal. In fact, many DNS spoofing and hijacking attacks occur in the terminal , LAN or ISP. Even if the DNS server is well protected, it cannot resist such attacks. Alharbi proposed a DNS poisoning attack on the client-side. This attack can bypass the existing DNS defense mechanism because the target of the attack is the client's DNS cache. The defense mechanisms against the DNS server cannot close this attack.

2.3 Objectives

According to the discussion above, there are two pain points in the existing DNS defense mechanism. First, the existing DNS Security mechanism mainly focuses on server-side. Unable to resist attacks against clients. Second, there are some problems such as inflexible deployment and difficult promotion. The methods for solving these two key issues will be discussed in detail below.

First, the problem of attacks against DNS client needs to be resolved. The current defense mechanism mainly focuses on the DNS server but ignores the protection of the DNS client. Protecting the DNS client is another idea. The role of DNS is to provide clients with IP addresses of a domain name. The ultimate goal of most DNS attacks (DNS spoofing, cache poisoning) is to give clients a fake IP address. If the client has the ability to discover fake IP addresses, then these attacks can be closed. Therefore, we need a method to verify the DNS response on the client. By this method, any DNS response containing a fake IP address will be detected and a warning will be issued to the user in time. The goal of this method is to protect DNS security in the end. Even if the hacker bypasses the existing DNS defense mechanism, this method can still ensure that the client has the ability to distinguish the fake IP address. How to achieve this goal? A simple and efficient method is to check the DNS cache of the computer. We all know that when users access a domain name, they will first retrieve the DNS cache. If the cache is not found, the DNS request will be sent. After receiving the result, the result will also be stored in the cache. Therefore, as long as the user visits a domain name, the domain name and its corresponding IP will be cached. So, we can design a verification system to verify the DNS cache of the computer. Whenever a DNS record enters the DNS cache, the computer sends the record to the system, and the system verifies the record and sends the verification result to the computer. The communication between the computer and verification system is encrypted and signed to ensure the confidentiality and integrity of the communication. Note that If a domain name uses CDN, the IP address of the domain name is different in different regions. Therefore, the verification system must be able to work normally under CDN environment.

Second, the problem of incompatibility with DNS needs to be resolved. DNSECE, DNS Over Https, DNS Over Tls, and DNScrypt are slow to promote because they all modify the DNS protocol. This makes them incompatible with DNS, making it difficult to promote. In order to be compatible with DNS, the new method should not modify the DNS. If the verification system is completely independent with the DNS, this goal can be achieved. The client obtains the IP of the domain name by DNS, then the verification system verifies it. The verification system works independently with DNS and does not interfere with each other. It should be noted that after the client receives a DNS reply, it will be used before verification. The purpose is to provide security services as much as possible without affecting the user experience. If the user receives a forged DNS response from a hacker, the most likely case is that the IP address in the forged response is a phishing website. The hacker wants the user to enter the account and password on his phishing website and steal user's property. But it takes time for users to enter information. Therefore, if verification system can complete the verification process in a short time, the user can be warned before the user completes the information input. Therefore, the verification system must complete the verification process in a relatively short time.

Based on the discussion above, we propose a client-side protection method for DNS cache. We call this method DNS cache guardian(DCG). The main objectives of DCG are as follows:

- Provide protection for DNS at the client-side.
- Can work normally in CDN environment.
- Fully compatible with DNS.
- Complete the verification process in a short time.

3 DCG

3.1 Principle

As shown in Figure 2, DCG adopts CS architecture, including client (client will be used to refer to DCG client later) and server. The client monitors the DNS cache of the user computer in real time. Once a new record enters the DNS cache, the client will send this record to the server. The server verifies this record and returns the results to the client. Note that DCG and DNS system work independently and do not interfere with each other. As an independent application, the client can be installed separately or integrated in the security software or browser to provide users with pluggable DNS Security services.

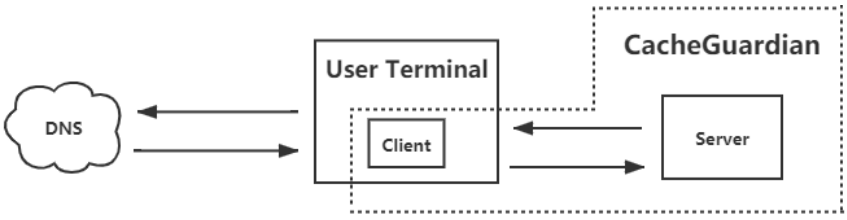

Figure 2: DNS and DCG

More details about DCG are shown in Figure 3. The terminal runs the client. The client monitors the DNS cache in real time. If a DNS record is cached in the terminal, the client will send the record to the server. The server will send the domain name in the record to queriers in multiple regions. These queries send query requests to DNS servers in this region and return the IP addresses found to the server. According to whether the IP returned by the queriers is consistent with the IP address in the cache record

sent by the client, the server determines whether the cache record sent by the client is correct. However, with the popularization of CDN technology, the IP address returned from the same domain name query may not be same in different regions. This paper puts forward a simple method to solve this problem. This method is to maintain a CDN legal IP table. The data in this table comes from the major CDN service providers.

The server includes transponder, validator, CDN legal IP validator, and multi-channel DNS querier. The transponder listens to client requests. The user's request is first sent to the transponder. After receiving the request, the transponder performs simple processing and then sends the data to the validator. The validator will generate the verification result and send the result to the transponder. The transponder processes the result and sends it to the client.

The validator is responsible for verifying the user's request (domain name and IP address). The basic idea is to send requests to multiple recursive parsers and compare the result with the original IP after receiving the result. However, if the domain name uses CDN technology, the above method cannot be used. Therefore, when verifying the user request, the validator first checks whether the IP belongs to the CDN legal IP. If not, the domain name does not use CDN. At this point, the domain name is sent to multiple recursive resolvers. If the IP belongs to the legal IP of the CDN, the verification is passed. The validator relies on the CDN legal IP validator and multi-channel DNS querier to verify the user's request. The CDN legal IP verifier will judge whether the IP address in the user request is a legal CDN IP. The multi-channel DNS querier is responsible for sending the domain to queriers located in multiple regions. The querier sends DNS query to the local DNS recursive resolver, and returns the resolution result to the multi-channel DNS querier. It should be noted that the communication between client and server, and the communication between server and queries use encryption and signature technology to ensure the integrity and confidentiality of communication information.

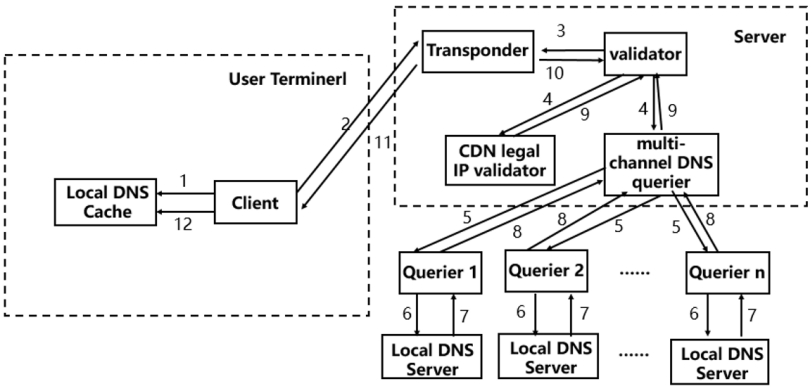

Figure 3: Architecture and workflow

3.2 Communication of Client and Serve

DCG provides users with DNS security service on the premise that the communication between the client and the server is secure. This section will detail the communication process between the client and the server.

3.2.1 Key Negotiation

The communication between the client and the server must be secure to prevent the attacker from monitoring the channel and forging the server's reply. Therefore, the communication needs to be encrypted

and signed to ensure the confidentiality and integrity of the data. This requires a session key to be negotiated between the client and the server. This key is used to encrypt the communication between the two parties. DCG uses RSA asymmetric encryption algorithm for session key negotiation. Note that the server's public key pub_s was already stored on the client when the client was installed. The process of key negotiation is shown in Figure 4.

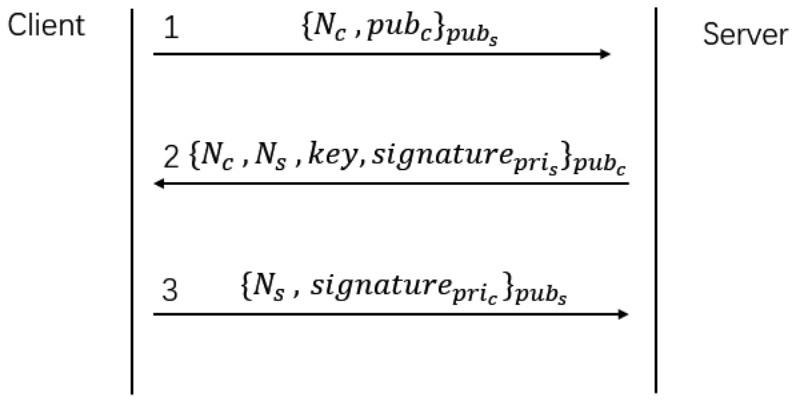

Figure 4: The process of key negotiation

First, the client sends $\{N_c, pub_c\}_{pub_s}$ to the server. N_c is the challenge random number generated by the client, and pub_c is the public key of client. The random number is to prevent replay attacks. The communication parties judge whether they have suffered a replay attack by comparing the received random numbers with the original ones. $\{N_c, pub_c\}_{pub_s}$ is the result of encrypting N_c and pub_s with the server public key pub_c.

Second, the server received the data sent by the client. The server uses its private key to decrypt the data to obtain N_c and pub_c. The server generates a random number N_s and a session key. Then, the server uses its private key pri_s to sign N_c, N_s, and key, and the signature result is $signature_{pri_s}$. Finally, the server uses the client public key pub_c to encrypt N_c, N_s, key and $signature_{pri_s}$, and send the result $\{N_c, N_s, key, signature_{pri_c}\}_{pub_s}$ to the client.

Third, after receiving the data, the client uses its private key to decrypt it to obtain N_c, N_s, *key* and $signature_{pri_s}$. The client uses pub_s to verify the $signature_{pri_s}$. Then check whether the random number N_c changes. If the above process fails verification, the client will immediately interrupt this negotiation. If the verification is passed, the client obtains the session key. Then, the client signs N_s with pri_s, and the result is $signature_{pri_c}$. Next, the client uses pub_c to encrypt N and $signature_{pri_c}$. Then send $\{N_s, signature_{pri_c}\}_{pub_s}$ to the server. Finally, after receiving the data, the server uses its private key to decrypt the data, and then verifies the digital signature and N_s. Similarly, if the authentication fails, the server will terminate this negotiation. If the verification is passed, the entire key agreement process is introduced.

After the above process is completed, the client and the server negotiate a session key. In the subsequent communication process, the client and server will use the session key to encrypt the communication. Now explain why not use RSA to communicate directly. Because RSA is an asymmetric cryptographic algorithm, its encryption and decryption process takes a long time. So only use RSA for key negotiation to get a symmetric key. The subsequent communication uses symmetric keys for encryption and decryption, and the speed will be improved. The session key will be used for a period of time. In DCG, this time is 30 minutes. In other words, every 30 minutes, the client will renegotiate a new session key with the server.

3.2.2 Data transmission

After the key negotiation between the client and the server, a session key is obtained. In the subsequent data transmission process, the key will be used to encrypt the data. The communication process is shown in Figure 5:

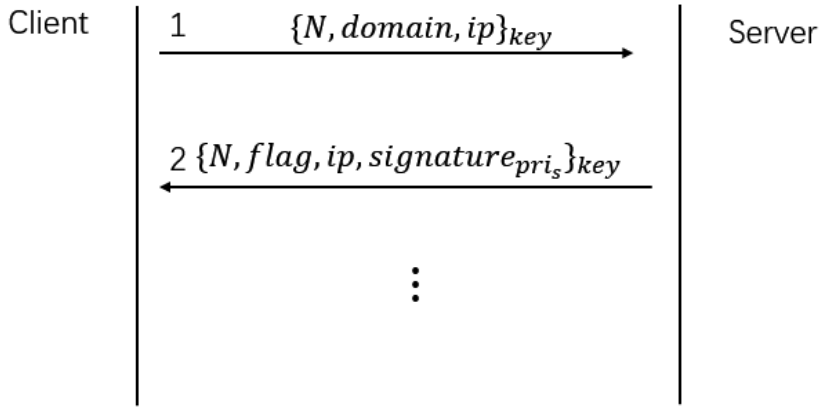

Figure 5: The process of data transmission

The data format of request of the client is $\{N, domain, ip\}_{key}$, where N is a random number, domain is the domain name, and ip is the IP address of the domain name. The data will be encrypted by the key. The role of random number N is to prevent replay attacks.

After receiving the request, the server uses the *keyto* decrypt message. Then verify the domain name and IP address. After the verification is completed, the server generates a verification result flag. The value of $flag$ is 0 or 1, 1 means that the verification passed, and 0 means that the verification failed. Regardless of whether the verification is passed or not, the server will generate ip. When the verification is successful, ip is the IP address in the client request. If the verification fails, ip is the correct IP address of the domain name. Then, the server uses its private key to sign N, $flag$, and ip, and the signature result is $signature_{pri_s}$. Finally, the server uses key to encrypt, $flag$, ip and $signature_{pri_s}$, then sends it to the client. After receiving the reply, the client first decrypts using the *key*, and then verifies the signature and random number in sequence. If the verification fails, the client discards the reply and thinks the server sends the request again. If the verification is passed, the client checks the value of $flag$. If $flag = 1$, it means that the verification of domain is passed. Otherwise, the verification of domain fails, and the client immediately warns the user and feeds back the ip to the user.

3.3 Implementation

3.3.1 Client

The function module of the client is shown in Figure 6. The client mainly includes three function modules, cache monitoring, alarm and communication.

The objects of monitoring are hosts file, browser DNS cache and system DNS cache. Therefore, the cache monitoring function module is divided into three sub-modules, which are used to monitor the above three objects. Reading the hosts file and the system DNS cache is relatively simple. Take the Windows system as an example. The hosts file is usually located in "C:\\Windows\System32\drivers\etc\hosts", and just read the file directly. The system DNS cache can be obtained through the command "ipconfig/displaydns". Browser DNS cache is more difficult to read. Chrome browser can read DNS cache

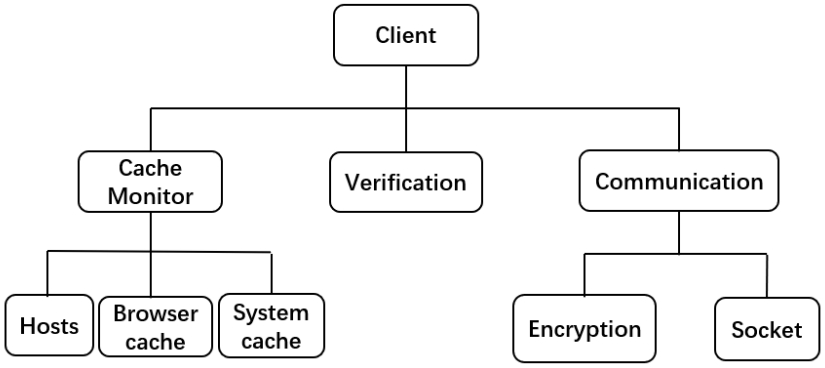

Figure 6: The function module of client

through url: "chrome://hnet-internals/♯ dns". Firefox and IE browsers do not provide an interface for viewing the DNS cache, so they cannot obtain their DNS cache. But this problem has no effect, because the entry into the browser DNS cache will also enter the system DNS cache. Algorithm 1 shows the workflow of this module in detail.

The function of the verification module is to take actions according to the verification result of the server. For example, if the result returned by the server is that the verification fails, the verification module will immediately warn the user. The module is very simple, so no detailed algorithm description is provided. The communication module is mainly used for communication between the client and the server. The communication module has two sub-modules, namely encryption and socket. The encryption sub-module is used to encrypt, sign, and decrypt communication data. The Socket submodule is used to transmit encrypted data. Algorithm 2 describes the workflow of the communication module in detail.

Algorithm 1 cache monitor

Input: hostsPath, showCacheUrl
Output: cacheRecord
$preHosts \leftarrow$ readFile($hostsPath$); $preBrowserCache \leftarrow$ readPage($showCacheUrl$)
$preSystemCache \leftarrow$ os.execute($'ipconfig/displaydns'$)
while ture **do**
  $hosts \leftarrow$ readFile($hostsPath$); $browserCache \leftarrow$ readPage($showCacheUrl$)
  $systemCache \leftarrow$ os.execute($'ipconfig/displaydns'$)
  $cacheRecords \leftarrow records$ which is in $hosts$,$browserCache$ and $systemCache$ but not in $preHosts$,$preBrowserCache$ and $preSystemCache$.
  Output($cacheRecords$)
  $preHosts \leftarrow hosts$; $preBrowserCache \leftarrow browserCache$;$preSystemCache \leftarrow systemCache$;
end while

3.3.2 Server

The function module of server is shown in Figure 7. The server mainly includes four function modules.

The Transponder module is mainly used for communication between the server and the client. The process of Transponder and multi-channel DNS querier is similar to Algorithm 2, so no detailed algorithm description is provided. This module contains two sub-modules: encryption and socket. The encryption sub-module is used to encrypt, sign, and decrypt data. The Socket submodule is used to

Algorithm 2 communication

```
Input: cacheRecord, key, privateKey
Output: the result(flag,ip) of verification
start: N ← random()
ciphertext ← encryption.signAndEncrypt( N, cacheRecord, key, privateKey)
socket.sent(ciphertext)
respond ← socket.listen()
if N and signature verification failed then
    goto start
else
    result ← encryption.getFlagAndIP(ciphertex)
end if
Output(result)
```

transmit encrypted data. The validator module is very important, and algorithm 3 describes the workflow of this module in detail. It mainly relies on CDN legal IP validator and multi-channel DNS querier to verify the request of users. The function of the CDN legal IP validator module is to detect whether the IP address is a legal IP address of the CDN. The dataset of CDN legal IP is stored in the database, and the submodule SQL provide database services. This module only involves database queries, so no algorithm description is provided. The multi-channel DNS querier communicates with the queriers. This communication process is similar to the communication between the client and the server. The module also has two sub-modules: encryption and socket. The encryption module guarantees the confidentiality and integrity of the communication, and the Socket module is used for data transmission.

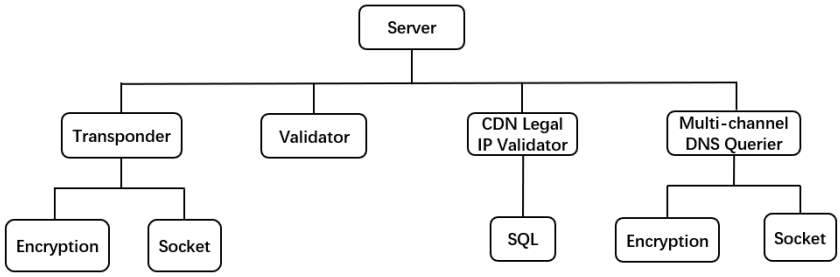

Figure 7: The function module of server

4 Evaluation

4.1 Experiment setup

The test environment topology is shown in Figure 8. The environment of experiment includes a target host, a hacker host, and a server. The target host and the hacker host are in the same LAN, and the IP addresses are 192.168.100.20 and 192.168.100.10. The IP address of the LAN gateway is 192.168.100.2. The operating system of the target host is Windows 7, and the network bandwidth is 100M. The target host runs the DCG client. The operating system of the hacker host is Kali Linux, and the network bandwidth is 100M. The server is a VPS, the operating system is Ubuntu, and the network bandwidth is 4M. The DCG serverruns on this VPS. The DNS recursive resolver uses a public resolver whose IP address is 114.114.114.114.

Algorithm 3 validator

Input: domain,ip
Output: $flag$,$correctIP$
$flag \leftarrow 0$
sent *domain* and ip to CDN legal IP validator and multi-channel DNS querier and get the results $CDNip$ and $querierIp$ respectively
$correctIP \leftarrow$ the most frequent IP address in $querierIp$
if $CDNip! = null$ **then**
  $flag \leftarrow 1$
  $correctIP \leftarrow ip$
  if $ip \leftarrow ==$ the most frequent IP address in $querierIp$ **then**
    $flag \leftarrow 1$
    $correctIP \leftarrow$ the most frequent IP address in $querierIp$
  else
    $flag \leftarrow 1$
    $correctIP \leftarrow$ the most frequent IP address in $querierIp$
  end if
end if
Output($flag$, $correctIP$)

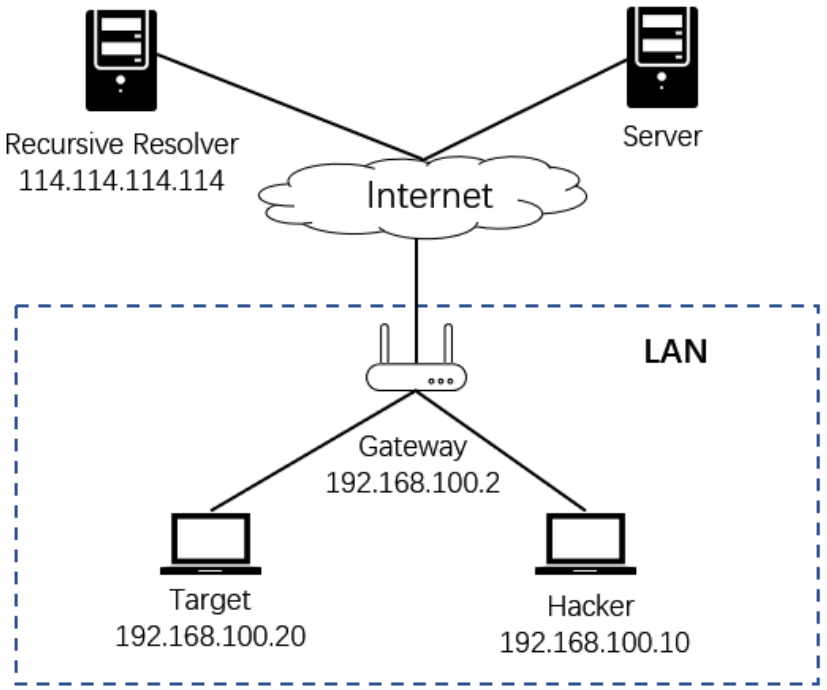

Figure 8: The environment of experiment

4.2 Analysis of experimental results

4.2.1 Attack simulation

We constructs a scenario of DNS spoofing attacks in a local area network to test whether DCG can work effectively. The attacker first uses the ettercap tool to perform ARP spoofing attacks. As shown in Figure 9, the attacker uses ARP spoofing to change the MAC address of the gateway in the ARP cache of the target host to the attacker's MAC address. In this way, the attacker pretends to be a gateway and hijacks all the traffic of the target host.

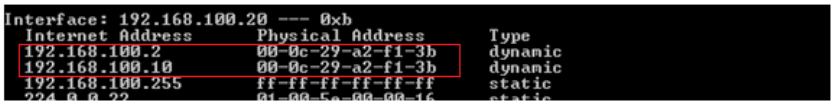

Figure 9: The ARP cache of target host

The domain names used for testing are www.gzhu.edu.cn and www.taobao.com. As shown in Figure 10, the correct IP addresses of these domain names are 58.205.213.52 and 42.236.122.10, respectively. www.gzhu.edu.cn does not use CDN, but www.taobao.com uses CDN. Table 1 compares DCG with other similar works. These works will be described in detail in Section 5.

Figure 10: Correct IP address

The attacker listens to the DNS requests of target host. If the target host sends a DNS request for the domain name (www.gzhu.edu.cn), the attacker will forge a DNS reply and send it to the target host. As shown in Figure 11, the IP address in the fake DNS reply was replaced with the attacker's IP address (192.168.100.10).

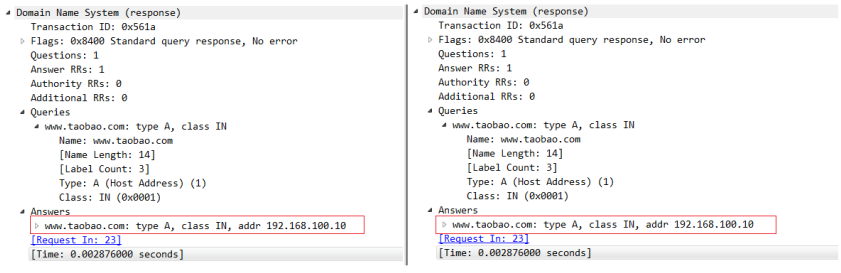

Figure 11: Forged DNS reply

After receiving the forged response, the target host will store the result in the DNS cache. As shown in Figure 12, the wrong result has entered the DNS cache of the target host. Note that the result of the domain name (www.taobao.com) is positive.

The DCG client is running on the target host. Therefore, when these two entries enter the DNS cache, the client has already detected and sent them to the server. After receiving the verification result from the server, the client will immediately warn the user if it finds an entry that fails verification. As shown

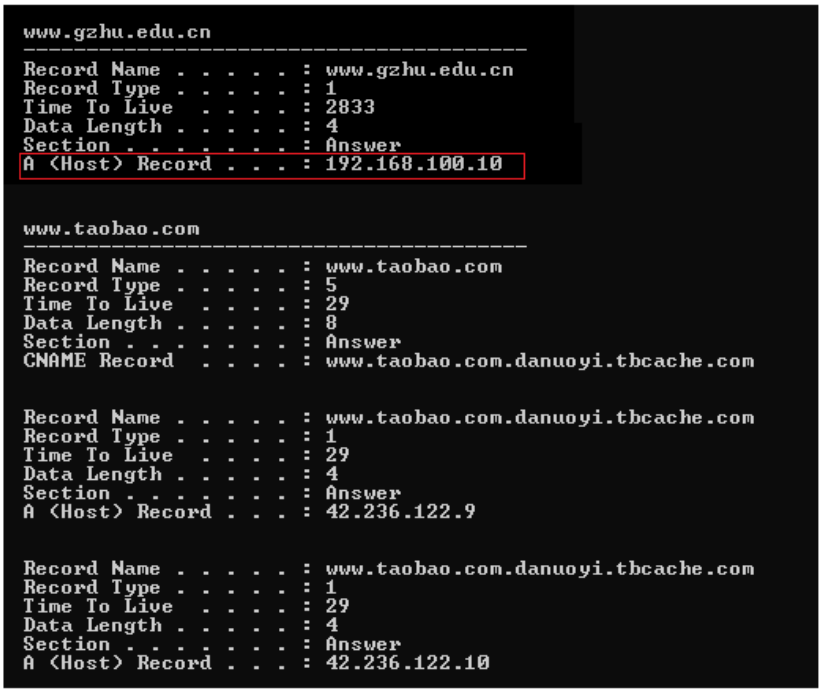

Figure 12: DNS cache of target host

in Figure 13, the client successfully detected the wrong entry and returned the correct IP address. Note that the domain name (www.taobao.com) was successfully verified, and the domain name used a CDN.

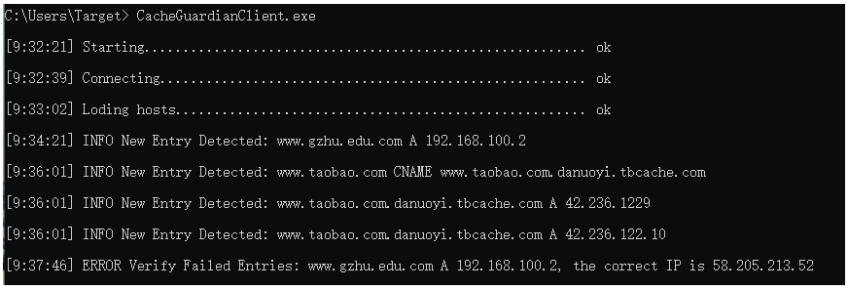

Figure 13: Output information of client

Experimental results show that DCG has basically achieved its design goals. First of all, DCG successfully detected the fake IP address, and provided DNS security services for the client. Second, as can be seen from the experimental topology, DCG is completely independent of the DNS system. No modification is required to the existing DNS system. Finally, DCG successfully verified the domain name (www.taobao.com) under the CDN environment. This shows that DCG can work normally in the CDN environment.

4.2.2 Time cost analysis

We measure the time taken by the components of this method and the communication between them. The specific items of measurement are shown in Table 2:

Table 1: Compare the DCG with other related works

Author	Protect DNS on Client-side	Work Normally with CDN
Our	Yes	Yes
Aksutov et al. [12]	Yes	No
Gastellier et al. [5]	Yes	No
Klein et al. [8]	No	Yes
Tzakikario et al. [16]	No	Yes

Table 2: Measurement items

client	processing time t1
server	server processing time t2
Communication	time between client and server t3
Communication	time between server and querier t4
Communication	time between querier and Recursive Resolver t5

The measurement items are mainly divided into two parts: processing and communication.

The processing part includes the client and the server. The main actions of the client include: monitoring the DNS cache in real time, encrypting the newly added cache entry and sending it to the verification server, receiving the server response data and decrypting and executing the response action. The time taken for this part of the action is t1. The main actions of the verification server include: decrypting the data, retrieving the legal CDN list, merging the querier, generating the verification result and signing the encryption, and returning the verification result. This part of the action takes time t2.

The communication part is divided into two parts: encrypted and unencrypted. The encrypted part is the communication between the client and the server and the communication between the server and the querier, and the time spent is t3 and t4, respectively. This part needs to encrypt and sign the communication process to ensure the confidentiality and integrity of the data. The unencrypted part is the communication between the querier and the recursive parser, which takes t5. Note that the time in this part is the round trip delay. This experiment will simulate the time spent measuring each item separately to simulate the total time T spent by the entire method.

The time spent by each measurement item is shown in figure 14, the verification client processing part cost t1 is 158 ms, and the verification server processing cost t2 is 268 ms. The round-trip delay t3 between the client and the server is 210ms. The communication time of the verification server is divided into a round trip delay t4 between the verification server and the querier and a round trip delay t5 between the querier and the recursive parser, which are 212ms and 156ms, respectively. From the data, most of the time is spent on communication. Therefore, the performance of this method mainly depends on the performance of the network.

Figure 15 shows how much time is spent on the domain name to be verified when using CDN and not using CDN. If the domain name to be verified uses a CDN, the verification takes less time and only takes 636ms. This is because if a CDN is used, when the IP address of the domain name to be verified appears in the list of valid CDN, the verification server will directly pass it. The time-consuming parts of t4 and t5 are avoided.

In summary, this method can verify the domain name in about 1s, and make corresponding warnings.

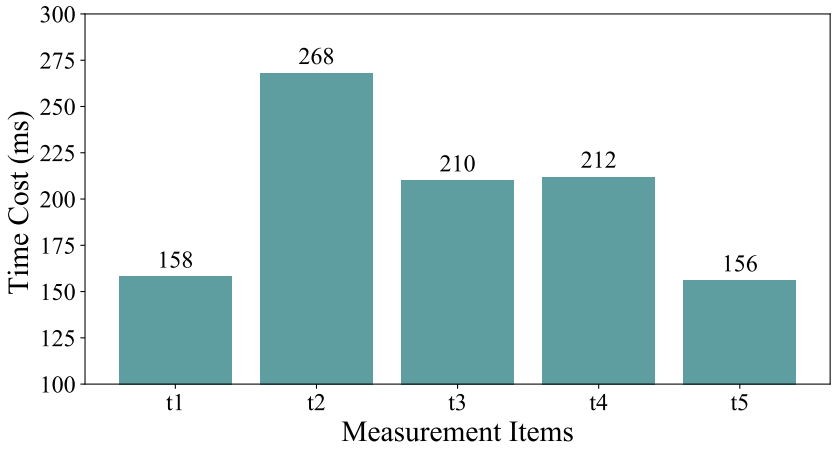

Figure 14: Time costs of items

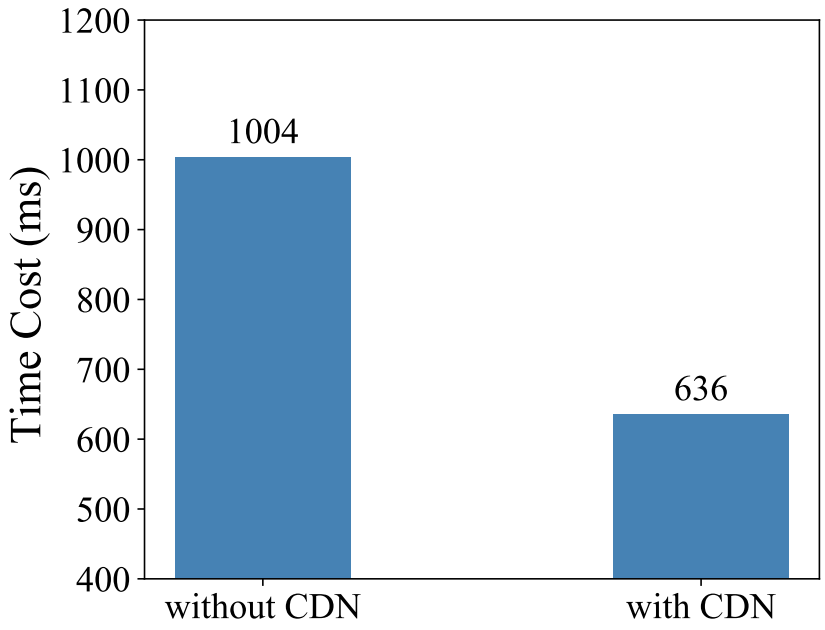

Figure 15: Time costs of with CDN and without CDN

The experimental results show that DCG basically achieves the expected goal.

5 Related work

Similar to the ideas of DCG, there are some studies on client-side protection of DNS security. For example, Aksutov et al. [12] proposes a method to detect DNS spoofing attacks on the client. Every time the client sends a DNS request, it sends an additional encrypted request to a trusted DNS server. The client compares the two results to determine whether the received IP address is a fake IP. This method requires additional software to be installed on a trusted DNS server. This method is simple and efficient, but does not consider the impact of CDN. Gastellier et al. [5] proposes a method for detecting DNS

spoofing on the client. The main idea is that when the client sends a DNS request to the simulated recursive resolver, it will additionally send the same request to multiple third-party recursive resolvers (such as OpenDNS, Google DNS, etc.). The client then compares the multiple response received and compares whether the IP address is consistent to determine whether the received IP address is false. Due to the wide application of CDN, IP addresses of the same domain name in different regions may be different. Therefore, this method does not work normally in the actual environment. Klein et al. [8] proposes a method for detecting DNS spoofing, which uses the whois command to perform a reverse query. This method also does not work properly in the CDN environment. Tzakikario et al. [16] proposes a DNS traffic authentication method. This method is used to authenticate DNS clients. Its purpose is to counter DDoS attacks against DNS servers, and it cannot solve DNS spoofing attacks against clients.

6 Discussion

This article proposes a method that can protect the DNS at the client-side. This method gives the DNS client the ability to identify false IP addresses. This method can solve DNS spoofing attacks and DNS cache poisoning attacks, because the ultimate purpose of these attacks is to allow users to access fake IP addresses. Any fake IP address that does not correspond to a certain domain name will be detected.

The basis of the security of the method proposed in this paper is to verify the security of the server. The server must implement strict security protection to prevent hackers from invading the verification server. The server is also at risk of DDoS attacks. An attacker can launch a DDoS attack on the server, making it incapable of working, and thus making this method unable to provide security services. Therefore, the server must formulate a strategy to deal with DDoS attacks. In addition, providing redundant backup is also needed to ensure that the backup server can continue to provide services when the server is down.

The method proposed in this article cannot deal with domain hijacking attacks and DDoS attacks. Domain hijacking generally refers to an attacker modifying the company's registered domain name record by impersonating the original domain name in an E-MAIL manner, or transferring the domain name to another organization, adding the domain name record to the designated DNS server after modifying the registration information, allowing the original domain name points to another IP server. When DCG verifies whether the IP address is the legal IP of a certain domain name, the basic assumption is that the authoritative DNS server can provide the correct IP address, so it cannot deal with such attacks. In addition, if the DNS server is hacked, the DNS record file is directly modified (for example, the hacker changed the A record to a fake IP address). DCG also can't deal with this kind of attack. DDoS attacks are a common DNS attack method. By controlling a large number of bots to send a large amount of data to a DNS server, the target server's resources (computing resources, network resources, etc.) are quickly exhausted, making the DNS server unable to provide services normally. DCG cannot deal with DDoS attacks. The main goal of DCG is to provide security for users at the client-side. Attacks against the DNS server are beyond the scope of DCG. In fact, the existing DNS security mechanisms (DNSSEC, DNS over HTTPs, DNS over TLS, etc.) are unable to respond to the above attacks. More seriously, because these mechanisms use encryption technology, this may make DDoS attacks against DNS servers easier.

7 Conclusion

We analyze the existing security problems of DNS, summarizes the existing defense mechanism of DNS, and proposes a Client-side Protection Method for DNS Cache, to improve DNS security. This method can defend against DNS spoofing attack and DNS cache poisoning attack at the client-side. Moreover, this method is compatible with CDN mechanism and provides users with a pluggable DNS security

service without changing the existing DNS system. Finally, it can be applied to browsers or anti-virus software, providing DNS security services for their users. And this method can also be developed into independent client software. Users only need to download and install it.

References

[1] F. Alharbi, J. Chang, Y. Zhou, F. Qian, Z. Qian, and N. Abu-Ghazaleh. Collaborative client-side dns cache poisoning attack. In *Proc. of the 14th IEEE Conference on Computer Communications (INFOCOM'19), Paris, France*, pages 1153–1161. IEEE, April-May 2019.

[2] S. Ariyapperuma and C. J. Mitchell. Security vulnerabilities in dns and dnssec. In *Proc. of the 2nd International Conference on Availability, Reliability and Security (ARES'07), Vienna, Austria*, pages 335–342. IEEE, April 2007.

[3] B. Braun. *Investigating DNS Hijacking Through High Frequency Measurements*. PhD thesis, University of California San Diego, 2016.

[4] F. Denis and Y. Fu. Dnscrypt. `https://dnscrypt.info/` [Online: accessed on May 20, 2020], December 2015.

[5] S. Gastellier-Prevost, G. G. Granadillo, and M. Laurent. A dual approach to detect pharming attacks at the client-side. In *Proc. of the 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS'11), Paris, France*, pages 1–5. IEEE, February 2011.

[6] P. Hoffman and P. McManus. Dns queries over https (doh). `https://tools.ietf.org/html/rfc8484` [Online; accessed on May 20, 2020], June 2018. IETF Internet-Draft (work in Progress).

[7] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for dns over transport layer security (tls). `https://tools.ietf.org/html/rfc7858` [Online; accessed on May 20, 2020], May 2016. IETF RFC 7858.

[8] A. Klein and Z. Golan. System and method for detecting and mitigating dns spoofing trojans. US Patent 8,266,295, September 2012.

[9] A. Klein, H. Shulman, and M. Waidner. Internet-wide study of dns cache injections. In *Proc. of the 12th IEEE Conference on Computer Communications (INFOCOM'17), Atlanta, Georgia, USA*, pages 1–9. IEEE, May 2017.

[10] Z. Kong and X. Jiang. Dns spoofing principle and its defense scheme. *Computer Engineering*, 36(3):125–127, 2010.

[11] H. Li, H. Ma, L. Haopeng, Z. Huang, X. Yang, K. Li, and H. Wang. Blockchain-based domain name resolution system. US Patent App. 15/768,833, May 2019.

[12] A. A. Maksutov, I. A. Cherepanov, and M. S. Alekseev. Detection and prevention of dns spoofing attacks. In *Proc. of the 1st Siberian Symposium on Data Science and Engineering (SSDSE'17), Novosibirsk, Russia*, pages 84–87. IEEE, April 2017.

[13] P. Mockapetris and K. J. Dunlap. Development of the domain name system. *ACM SIGCOMM Computer Communication Review*, 18(4):123–133, August 1988.

[14] A. Rigoni, I. N. Fovino, S. Di Blasi, and E. Casalicchio. Worldwide security and resiliency of cyber infrastructures: The role of the domain name system. In *Proc. of the 2nd Worldwide Cybersecurity Summit (WCS'11), London, UK*, pages 1–5. IEEE, June 2011.

[15] D. SECUREWORKS. Dns cache poisoning-the next generation. `https://www.secureworks.com/blog/dns-cache-poisoning` [Online: accessed on May 20, 2020], March 2011.

[16] R. Tzakikario, D. Touitou, and G. Pazi. Dns anti-spoofing using udp, November 2009.

[17] H. Yang, E. Osterweil, D. Massey, S. Lu, and L. Zhang. Deploying cryptography in internet-scale systems: A case study on dnssec. *IEEE Transactions on Dependable and Secure Computing*, 8(5):656–669, April 2010.

Author Biography

Yan Zhao received his BS degree in Computer Science from Zhengzhou University (ZZU) in 2014 and is pursuing his MS degree in Computer Science at the Guangzhou University. His research interests are in the areas of network security and industrial control technology.

Ning Hu received the Ph.D. degree from the National University of Defense Technology (NUDT), China. He is currently a Professor of the Cyberspace Institute of Advanced Technology, Guangzhou University. He has published over 30 papers in journals and conferences. His current research interests include network security and Internet of Things. E-mail: huning@gzhu.edu.cn.

Chi Zhang received his BS degree in Communications Engineering from China University of Petroleum (UPC) in 2014 and is pursuing his MS degree in Computer Science at the Guangzhou University. His research interests are in the areas of network security and industrial Anonymous communication.

Xinda Cheng received his BS degree in Computer Science from Shandong University(SDU) in 2015 and is pursuing his MS degree in Computer Science at the Guangzhou University. His research interests are in the areas of network security and industrial Anonymous communication.