Security Analysis and Solution for Thwarting Cache Poisoning Attacks in the Domain Name System

Ramzi Bassil    Roula Hobeica    Wassim Itani*  
Department of Electrical and Computer Engineering  
American University of Beirut  
Beirut 1107 2020, Lebanon  
{rtb01,rah70,csg04,ayman,chehab}@aub.edu.lb

Cesar Ghali    Ayman Kayssi    Ali Chehab  
*Department of Electrical and Computer Engineering  
Beirut Arab University  
Beirut 1107 2809, Lebanon  
w.itani@bau.edu.lb

ABSTRACT  
The Domain Name System is a crucial part of the Internet’s infrastructure, as it provides basic information that is vital for the proper operation of the Internet. The importance of DNS has caused it to be targeted by malicious attackers who are interested in causing damage and gaining personal benefits. Thus nowadays, DNS faces many security threats such as DNS spoofing and cache poisoning attacks. This paper presents S-DNS, an efficient solution for thwarting cache poisoning attacks in the DNS hierarchy. The contribution of the S-DNS protocol lies in: (1) decreasing the success probability of DNS spoofing and cache poisoning by preventing man-in-the-middle attacks, (2) providing a backward compatible and simple security solution with low computation and communication overheads, (3) targeting the different DNS query interaction models from iterative, recursive, and caching schemes, and (4) employing an efficient Identity-Based Encryption key management scheme that relieves the different DNS interacting entities from the burden and complexities of traditional public-key infrastructures.

Keywords  
DNS, DNS security, DNS cache poisoning, spoofing attacks, man-in-the-middle attacks, Identity-Based Encryption.

1. INTRODUCTION

For Internet users to access the web they should be able to identify the object they want to access. Internet hosts are identified by their hostnames and IP addresses [1]. The main task of the Domain Name System (DNS) is to provide the translation from hostnames to IP addresses, and thus it is considered the Internet’s directory. When a client needs to translate a hostname into an IP address, a series of events occur before receiving the response. First, the client side of DNS (resolver) is invoked by the requesting application, which gives it the hostname that needs to be translated. Then, this resolver sends a DNS query over UDP to the local DNS server. After a certain delay, the resolver receives a reply message providing the IP address of the queried hostname. This IP address is then passed to the application (eg. web browser), which will be able to access the host using its IP address. DNS uses a large number of servers distributed in a hierarchical fashion. The hierarchy is divided into three classes: root DNS servers, top-level domain (TLD) servers, and authoritative DNS servers. There are thirteen root DNS servers in the world, labeled A through M. TLD servers are responsible for top-level domains such as .com, .net, .edu, etc. Authoritative DNS servers are for each organization or entity that wishes to have publicly accessible hosts. Another type of servers is the local DNS servers. These servers are not considered to be part of the hierarchy but are very important for efficient DNS functionality.

The local server is close to the host and caches mappings in its local memory. DNS caching is an essential feature because it improves the performance of DNS and reduces the number of DNS messages. DNS servers store entries in their databases in the form of resource records (RRs).

This paper presents S-DNS, a security solution for thwarting cache poisoning attacks in the DNS hierarchy. The contribution lies in: (1) decreasing the success probability of DNS spoofing and cache poisoning by preventing man-in-the-middle attacks, (2) providing a backward compatible and simple security solution with low computation and communication overhead, (3) targeting the different DNS query interaction models from iterative, recursive, and caching schemes, and (4) employing an efficient Identity-Based Encryption (IBE) key management scheme that relieves the different DNS interacting entities from the burden and complexities of traditional Public-Key Infrastructure PKI.

The rest of the paper is organized as follows: In Section 2 we briefly describe the operation of the DNS protocol and the security vulnerabilities it suffers from. In Section 3 we present a survey of the main security protocols proposed for securing the DNS infrastructure. Section 4 discusses the feasibility of source port and transaction ID guessing attacks in DNS by providing an experimental timing and security analysis. Section 5 presents the S-DNS protocol design and describes the main cryptographic mechanisms employed in securing the DNS query/reply messages. In Section 6 we provide a description of a prototype implementation of the proposed protocol design and compare the processing and data transfer overhead between the S-DNS and the DNSSSEC protocol, while in Section 7 a high level comparison between all the proposed DNS security protocols is presented. Conclusions are presented in Section 8.

2. DNS SECURITY ISSUES

DNS faces many security problems and much effort was put to both protect and attack DNS. The most commonly encountered security threat is DNS cache poisoning which exploits the fact that DNS messages are sent in the clear and that neither DNS entries, nor DNS servers are authenticated [2]. Therefore, malicious users can sniff DNS queries and inject erroneous RRs in DNS servers. In order for this attack to succeed the attacker should successfully perform a man-in-the-middle attack where he intercepts a DNS request to the local server and directly replies with a spoofed DNS reply message to this same server, and this reply message has a significant possibility of being accepted as the attacker already knows the transaction ID (TxID) entry in the RRs which is used to identify DNS queries and replies, since he was able to intercept the request. Another attack is when the attacker guesses the TxID and source port entries for queries sent by other users, but this
attack is much less feasible as will be shown in Section 4. As a result of this attack, queries sent to this DNS server will receive the records injected by the attacker and will be redirected to wherever the attacker decides. The vulnerability of the DNS protocol is proved in [3]. Despite the use of security applications and phishing filters, DNS poisoning attacks still have a non-negligible degree of success. What makes matters worse is that security toolbars and phishing filters provide the user with confirmation that the malicious site is legitimate.

One of the earliest proposed solutions to resolve DNS vulnerabilities is the DNSSEC extension developed by the Internet Engineering Task Force (IETF) and published in 1997 [4]. This extension adds security measures to the original DNS protocol by using cryptography. In this respect, all DNS entries are digitally signed using asymmetric keys, which ensures data and server authentication as well as integrity [5]. As such, when the DNS local server receives the reply message, it must receive, along with the message, an additional record called RRSIG containing the record’s signature. To verify this signature, the local server must decrypt it using another received record which is the DNSKEY. Moreover, to make sure that the DNSKEY is valid, the local server hashes it and compares it to a DS record previously received from the parent server. This way, a man in the middle attack would fail since the attacker has no way of properly signing the required record, and as soon as the resolver detects the missing or incorrect signature, it will not use the provided information. Although DNSSEC has been around for more than ten years, it still has not been widely deployed because it adds an additional layer of complexity due to the increased computation required by public key cryptography mechanisms. Another problem faced by DNSSEC is the fact that nations have not agreed on “who owns the name servers”.

The proposed DNS security protocol presented in this paper provides a much enhanced performance when it comes to time and data overhead as compared to DNSSEC. This is because, as will be described in Section 5, S-DNS adds an additional RR for the query message and another RR for the reply message; a total of two additional RRs. On the other hand, DNSSEC adds four new RR types: resource record signature (RRSIG), DNS public key (DNS KEY), Delegation Signer (DS), and Next Secure (NSEC). Furthermore, the security of DNSSEC is based solely on the trust in the upper layer DNS servers. So, if a root server is compromised, the security of DNSSEC in the lower layer servers fed by this server will fail, and as a result the whole functionality of DNSSEC will collapse. Thus in order to be confident in the security level provided by DNSSEC, the upper layer servers must be trusted, which is not always the case. This is asserted by the attack described in [15] which necessitates the complete reform of the DNS infrastructure to prevent any fraud activity on PKI certificates. On the other hand, S-DNS relies on the trust and security of the IBE Private Key Generator (PKG) entity. Thus securing this entity provides robustness in the S-DNS system and avoids the reliance on a long chain of trust among the different DNS servers. Moreover, S-DNS circumvents the question “who owns the root servers”, rendering its possible adoption practical.

3. PREVIOUS WORK

An alternate method that could be used instead of DNSSEC is proposed in [6]. Using this method, inaccurate records including obsolete records can be detected and corrected. This is done by implementing a client-side system consisting of what is referred to as DoX peers which are in fact proxy servers. These peers communicate with each other, and perform consistency checks with each other. If a suspicious record is noticed, they consult the authoritative server. The basic reason behind the success of this method is the fact that it is very hard for an attacker to succeed in poisoning the records of several distributed servers at the same time. So by comparing records, DoX peers will find the incorrect record. A verification cache is added by DoX peers to each proxy, this means that whenever a certain record is verified, it is added to this cache, and propagated to other peers so that all peers are synchronized. This verified cache would not need to be re-verified, thus improving efficiency [7].

Employing a similar concept to that of DNSSEC is an enhancement to DNS security using the SSL infrastructure. The method presented in [2] gives hosts the freedom of using the added security or disregarding it if they find it to be more of an overhead than a benefit. In this method, instead of having all DNS entries in an authoritative server signed by the server’s zone, a DNS entry can be signed by its respective owner. For this purpose, an additional resource record type is added in the DNS server’s records. To check for the validity of the owner’s certificate, a trusted third party (certification authority) is used by the user. Resolvers with high security needs would ask for the signed RRs, while resolvers that do not care about security issues could ask for the regular RRs.

Another technique that disables cache poisoning attacks is Wildcard SECure DNS (WSEC DNS) [8]. This method uses the definition of wildcard domain names and TXT resource records. A wildcard domain name is a domain name having its initial label as the “*” character. For example, *.www.example.com is a wildcard domain, where “*” is interpreted as any valid combination of characters. The basic idea of this method is to use wildcard domains and add a random string instead of “*” to the queried domain name and still obtain a correct answer. The added strings would increase the randomness in the DNS queries thus making poisoning attacks much harder to complete.

Another method that is also compatible with the existing DNS service is presented in [9]. This approach provides a detection method for cache poisoning attacks that could be due to sniffing or even originating from a birthday attack on a recursive DNS server. This system, named Cache Poisoning Detection System (CPDS), employs an architecture composed of a recursive query manager, an iterative query manager, a database manager, a cache validation manager, as well as an alert manager, and is applied on the recursive server. The paper confirms that the security provided by CPDS is similar to that provided by DNSSEC. However, the CPDS method incurs much less overhead due to the absence of the computational load required by cryptography.

IBE is a cryptosystem where the public key can be any public string assigned to a specific entity such as a host name, an IP address, an email address etc. This scheme is a suitable replacement of the traditional Public Key Infrastructure [13]. In an IBE method, there is no need for a public key lookup phase from a trusted-keys repository since the encrypting entity can directly use the destination public string for encrypting the message content. In addition, any entity using the IBE method can send an encrypted message to another entity that does not possess a private key yet. Moreover, in IBE, an entity’s private key is refreshed periodically which eliminates the need for revocation lists.
4. DNS TIMING AND SECURITY ANALYSIS

In order to propose a security scheme that would effectively protect DNS from its current vulnerabilities, some relevant parameters were measured and studied using the tools given in [10]. To start with, the DNS Benchmark program was used. Within this tool, the two local DNS servers (in this case, the servers belonging to the ISP) as well as other publicly known major name servers such as servers belonging to “Level 3 Communications” were listed for testing. Response times from each of the tested DNS servers are recorded as shown in Figure 1.

As can be seen in Figure 1, DNS servers are ranked from the fastest response to the slowest. Note that the first two servers are those belonging to the local ISP. Red labels in the results represent the time it took to resolve cached entries, whereas green labels represent the time it took to resolve non-cached entries. Blue labels are an indicator of how well the DNS server being tested is connected to the (.com) name servers. Regarding the first ranked DNS server, the response times were as shown in Figure 2. These results are expected since the ISP’s local DNS server usually has a large cache, and is the closest to the user. From these results one can imply that in case a malicious attacker wants to tamper with DNS replies, he has a time window of the order of 0.1 seconds.

![Figure 1. DNS Benchmark Results.](image1)

![Figure 2. Results for Local ISP’s DNS Servers.](image2)

Using [11], other parameters were studied including the distribution of the 16-bit Source Port and TxID numbers. Figure 3 shows the distribution of the TxID numbers for 1000 queries over the range [1-65535]. This experiment was conducted under the Microsoft Windows 7 operating system. The same experiment showed a similar distribution for source port numbers.

According to the distribution results, it can be inferred that the Local DNS server queries contained diverse source port numbers and TxID numbers that make use of almost all possible values within the range, thus making it more immune against spoofing attacks based on guessing. In addition, measures were done to see the predictability of the source port number and TxID of a DNS query. Results for TxID predictability are shown in Figure 4, and similar results were obtained for source port predictability.

According to the predictability measures and the timing analysis, we can conclude that it is technically hard to achieve successful source port and TxID guessing attacks on the DNS queries traversing the network links between the client and the local DNS server. Moreover, the same analysis corroborates the fact that guessing attacks have a very narrow time window between the local DNS server and the rest of the DNS hierarchy. This renders man-in-the-middle attacks, where a malicious entity intercepts DNS packets, the most dangerous threat facing DNS.

![Figure 3. TxID Distribution.](image3)

![Figure 4. TxID Predictability.](image4)

5. SYSTEM DESIGN

This section presents S-DNS, the DNS security solution proposed in this work. The main solution objectives and contributions are listed below:

- Decreasing the success probability of DNS spoofing and cache poisoning by thwarting man-in-the-middle attacks and preventing them from gaining any advantage from intercepted DNS queries and replies. Based on the analysis presented in the previous section, TxID and source port guessing cannot be relied upon to lead to successful cache poisoning attacks. Thus, mitigating the risk of man-in-the-middle attacks should be the
natural target for achieving safe and reliable DNS hierarchy interaction.

- Providing a simple security solution that complies with the inherent simplicity of the DNS protocol design.
- Reducing the computational and communication overhead of the security protocol by focusing on a specific and significant DNS security problem which is represented by man-in-the-middle attacks. This gives the proposed solution major performance advantages over comprehensive public key-based security protocols such as DNSSEC.
- Achieving backward compatibility by allowing the smooth interaction of S-DNS and legacy DNS servers.
- Targeting the different DNS interaction models such as the dominant iterative query model (see Figure 5), the recursive query model, and the caching-based DNS scheme where the answer to a query is fetched from the cache of an intermediate server without traversing the hierarchy all the way down to the authoritative DNS server.
- Employing an efficient identity-based encryption key agreement scheme that relieves the different DNS interaction entities from the burden and complexities of traditional PKI hierarchies.

Figure 5 presents the main architectural entities of the iterative S-DNS interaction model. The system design consists of three main components:

1. The S-DNS requester: This is any entity that initiates a DNS query. The requester could be the client, the local DNS server in the iterative DNS model, or the local DNS server, the root DNS server, or the TLD DNS server in the recursive DNS model.
2. The S-DNS responder: This is any entity that responds to a DNS query. This responder could be a local DNS server, a root DNS server, a TLD DNS server, or an authoritative DNS server in both recursive and iterative query models.
3. IBE Private Key Generator (PKG): This is a trusted entity responsible of generating private keys to IBE communicating parties as implied by the IBE specifications.

5.2 The S-DNS Secret Decryption and MAC Generation Phase

This phase is executed at the responder. As the requester query reaches the responder DNS server, the encrypted S-DNS Secret is decrypted using the destination server’s private key. This key is periodically acquired by contacting and authenticating to the PKG entity as implied by the IBE specifications. This means that revocation is possible; whenever a server is deemed malicious the PKG would not renew its key and thus it will be revoked. The extracted random sequence is used as a symmetric key to generate a Message Authentication Code (MAC) of the reply DNS message. As is the case in the S-DNS Secret creation phase, the MAC value generated is added as an additional DNS resource record. In this way, an attacker intercepting the DNS reply message cannot maliciously modify the message content without being detected by the requester. For better utilization of the server resources, S-DNS supports the key reuse concept which saves the relatively complex public-key encryption and decryption operations by reusing the random shared S-DNS secret in multiple consecutive sessions between the DNS requester and responder.

In this sense, the DNS requester indicates via an explicit flag that it would like to reuse the shared random secret already established with the responder. This saves a random number generation and a public-key encryption operation on the requester side and a public-key decryption operation on the responder side. When employing the key reuse algorithm in this phase, the responder only needs to perform a simple MAC operation using the already established random sequence to ensure the integrity and authenticity of the DNS reply. The key reuse concept represents a tradeoff between performance and security as will be illustrated in Section 6. The operation of the S-DNS Secret decryption and

along with the public key of the destination server is fed to an IBE encryption algorithm to produce the S-DNS Secret data structure. Afterwards, the requester DNS server constructs the DNS query, creates an additional DNS RR with a custom type, and assigns the S-DNS Secret to the name field of this record. This procedure utilizes the fact that BIND [12], which is the name server software in the majority of DNS servers, has the ability to deal with new RR types. The execution steps of the S-DNS Secret creation phase and the structure of the new DNS query message are presented in Figures 6 and 7, respectively.

5.1 The S-DNS Secret Creation Phase

This phase is executed by the S-DNS requester. The main purpose here is to create an S-DNS Secret encrypted using the IBE public key of the next DNS server in the hierarchy. This public key is a well-known string based on the IBE specifications. This is illustrated as follows: a 128-bit randomly generated bit string
MAC generation phase and the structure of the DNS reply message are presented in Figures 8 and 9, respectively.

By employing the key reuse algorithm presented in Section 5.2. The session size refers to the number of consecutive DNS query/reply messages secured using the same S-DNS Secret.

On the other hand, the session size parameter has a major impact on the data transfer overhead between DNS requester and DNS responder servers, as illustrated in Figure 12. For instance, the data transfer overhead factor is 1.24 for a session size of 1, which is enhanced to 1.135 for a session size of 50 (see Figure 12).
Detailed analysis of these factors is beyond the scope of this paper.

Figure 12. Data Transfer Overhead Factor vs. Session Size.

7. PERFORMANCE COMPARISON
This section describes a high level comparison between S-DNS and all the other DNS security protocols surveyed in this work. Table 1 presents a comparison between DNS security protocols according to different criteria. Vertical cooperation between DNS servers means that TLDs communicate with root servers while authoritative servers communicate with TLDs. Horizontal cooperation means that servers from the same class communicate together, and those are usually local DNS servers. Protocols like DoX Peers and CPDS have a slow poisoning discovery rate because they act after the poisoning takes place and try to foil the attack, but by then the attacker could have compromised many legitimate users. Ease of deployment is a function of the additional equipment required to implement the protocol, so the less the requirements the easier the deployment.

8. CONCLUSIONS
This paper presented an efficient security protocol for preventing DNS cache poisoning attacks in DNS. The proposed protocol aims at decreasing the success probability of DNS spoofing and cache poisoning by preventing man-in-the-middle attacks, providing a backward compatible and simple security solution with low computation and communication overheads, targeting the different DNS query interaction models, and employing an efficient Identity-Based Encryption key management scheme. Last but not least, any DNS security solution has its drawbacks as well as its benefits, and the solution that would be widely accepted is the one that has the best combination of time of introduction, affordability, ease of implementation as well as security level.

Table 1. DNS Related Protocols Comparison

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Server Cooperates with</th>
<th>Encryption Overhead</th>
<th>Rate of Poisoning Discovery</th>
<th>Ease of Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-DNS</td>
<td>IBE PKG</td>
<td>Medium</td>
<td>Fast</td>
<td>Medium</td>
</tr>
<tr>
<td>DNSSEC</td>
<td>DNS Servers (vertically)</td>
<td>High</td>
<td>Fast</td>
<td>Low</td>
</tr>
<tr>
<td>DoX Peers</td>
<td>DNS Servers (horizontally)</td>
<td>None</td>
<td>Slow</td>
<td>High</td>
</tr>
<tr>
<td>SSL DNS</td>
<td>CA</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>WSEC DNS</td>
<td>DNS Servers (vertically)</td>
<td>None</td>
<td>Slow</td>
<td>High</td>
</tr>
<tr>
<td>CPDS</td>
<td>DNS Servers (horizontally)</td>
<td>None</td>
<td>Slow</td>
<td>Slow</td>
</tr>
</tbody>
</table>

REFERENCES