A Client Based DNSSEC Validation System with Adaptive Alert Mechanism Considering Minimal Client Timeout

Yong JIN(a), Member, Kunitaka KAKOI(b), Nonmember, Nariyoshi YAMAI(c), Member, Naoya KITAGAWA(d), Nonmember, and Masahiko TOMOISHI(e), Member

SUMMARY The widespread usage of computers and communication networks affects people’s social activities effectively in terms of intercommunication and the communication generally begins with domain name resolutions which are mainly provided by DNS (Domain Name System). Meanwhile, continuous cyber threats to DNS such as cache poisoning also affects computer networks critically. DNSSEC (DNS Security Extensions) is designed to provide secure name resolution between authoritative zone servers and DNS full resolvers. However high workload of DNSSEC validation on DNS full resolvers and complex key management on authoritative zone servers hinder its wide deployment. Moreover, querying clients use the name resolution results validated on DNS full resolvers, therefore they only get errors when DNSSEC validation fails or times out. In addition, name resolution failure can occur on querying clients due to technical and operational issues of DNSSEC. In this paper, we propose a client based DNSSEC validation system with adaptive alert mechanism considering minimal querying client timeout. The proposed system notifies the user of alert messages with answers even when the DNSSEC validation on the client fails or timeout so that the user can determine how to handle the received answers. We also implemented a prototype system and evaluated the features on a local experimental network as well as in the Internet. The contribution of this article is that the proposed system not only can mitigate the workload of DNS full resolvers but also can cover querying clients with secure name resolution, and by solving the existing operation issues in DNSSEC, it also can promote DNSSEC deployment. key words: DNSSEC, DNSSEC validation, client based DNSSEC validation, DNSSEC failure alert, and DNSSEC timeout alert

1. Introduction

The Internet has been developed into an indispensable information infrastructure for human’s social activities and it makes computers all over the world connect and communicate with each other. In order to identify the computers connected to the Internet, one or more IP addresses need to be assigned to each computer and they are also used for communications. However, IP address is not human-friendly identifier for computers, therefore domain names are well used instead. Since the domain names need to be translated to corresponding IP addresses before communication with each other in the computer networks, an effective distributed database system named DNS (Domain Name System) [1], [2] is widely used and deployed.

By convention, DNS protocol was considered as a secure communication protocol so that each computer connected to the Internet trusted the results of name resolution received from its corresponding DNS full resolvers. With the development of research in DNS, it becomes successful for attacker to plant poisons (wrong information) into DNS full resolvers using clever tricks. Consequently, malicious DNS response such as wrong IP address that belongs to a phishing site can be replied to the querying client by the poisoned DNS full resolver and consequently the querying client will be directed to malicious sites. A typical case is DNS Kaminsky attack [3] which is practically using query ID in DNS message and make the DNS full resolvers accept disguised DNS responses. Although DNS Kaminsky attack was known long ago, it was considered as non-realistic. However, in 2008, Dan Kaminsky reported that he has successfully raised the possibility of the attack and the risk has become reality. To solve the problem DNSSEC (DNS Security Extensions) protocol [4], [5] has attracted much attention.

DNSSEC has been developed and standardized for over ten years but its deployment rate is still low over the Internet. One of the important reasons is high administrative cost of key management on authoritative zone servers. In DNSSEC, the integrity of DNS response is guaranteed by using Public Key Cryptography [6] and digital signature, thus authoritative zone servers have to maintain private and public key pairs periodically and correctly. In addition to the high maintenance cost, once the rotation of the key pairs fails then all the zones managed by the authoritative zone server will be failed with DNSSEC validation and it will result with complete name resolution failure on querying clients. Another important reason is that in general DNSSEC validation is performed on DNS full resolvers in order to practically use its cache function, therefore the DNSSEC validation process will significantly increase the workload of DNS full resolvers which causes the decrease of name resolution performance [7]. Furthermore, when DNSSEC validation is failed on DNS full resolvers, the querying clients cannot receive any answer except failure message, therefore the querying clients cannot perform the following communications until the DNSSEC validation problem is solved [8]. Finally, querying clients are not covered by DNSSEC with
secure name resolution since DNSSEC validation is mainly done on DNS full resolvers. Therefore, besides the link between DNS full resolvers and querying clients, in many network environments such as free Wi-Fi spots, querying clients cannot use DNSSEC due to that the default DNS full resolvers do not support DNSSEC.

In order to solve the existing operational issues in the current DNSSEC, we propose a client based DNSSEC validation system with alert mechanism considering not only the DNSSEC validation failure but also its timeout. In the proposed system, DNSSEC validation process is moved from DNS full resolvers to each querying client and alert messages indicating DNSSEC validation failure as well as name resolution timeout caused by DNSSEC validation will be displayed to the users to make them determine whether trust the answer or not. By using the proposed system, first, the workload increase caused by DNSSEC validation on DNS full resolvers can be mitigated, then more importantly, a querying client not only can use DNSSEC with secure name resolution but also can practically control the DNS responses based on its DNSSEC validation results. We also implemented a prototype system and evaluated the proposed features in a local experimental network as well as in the Internet. Based on the evaluation results, we confirmed that the proposed system worked well as we expected and it can contribute to solve the existing operational issues in the DNSSEC as well as to promote the deployment of DNSSEC over the Internet.

2. Overview of DNSSEC and the Operational Issues

The typical security concerns in conventional DNS is cache poisoning attacks to DNS full resolvers. In most organizations, several official DNS full resolvers are constructed to provide name resolution services for their internal computers effectively. Attackers target at those DNS full resolvers and make them cache and provide malicious DNS information to querying clients. In order to provide secure name resolution service, DNSSEC protocol has been designed and it is also a solution for cache poisoning attacks. DNSSEC uses Public Key Cryptography based digital signature technology and DNS full resolvers can validate the authenticity and integrity of the received DNS responses before saving them in cache. Therefore DNSSEC can mitigate the risk of DNS cache poisoning attacks.

2.1 Overview of DNSSEC

Basically DNSSEC provides secure name resolution between authoritative zone servers and DNS full resolvers. As shown in Fig. 1, first, the authoritative zone server calculates the hash value of the Resource Record set (RRset) using one-way hash function and encrypts the hash value using its private key to create the corresponding RRSIG record which is the digital signature of the RRset. Next, when the authoritative zone server receives a query from the DNS full resolver, it replies back the RRset and its correspond-

![Fig. 1 DNSSEC mechanism](image)

ing RRSIG record. On the other hand, when the DNS full resolver receives DNS response from the authoritative zone server with its corresponding RRSIG record, it calculates the hash value using the same one-way hash function for the received RRset and then decrypt the received RRSIG record using the public key of the zone. Note that the public key is published as DNSKEY record of the zone and also will be signed by the private key of the parent zone as its DS (Delegation Signer) record for providing trust train. Finally, the DNS full resolver checks if the calculated hash value is identical to the one decrypted from the RRSIG record. If the two hash values are identical then the DNSSEC validation is successful, otherwise vice versa.

2.2 Existing Operational Issues in DNSSEC

As described in the previous section, DNSSEC is a good option for DNS cache poisoning attacks and it is expected that wide spread of DNSSEC can reduce the risk of DNS cache poisoning attacks significantly. However, in terms of real operation, there are several existing issues hindering DNSSEC deployment. First, DNSSEC based name resolution involves big DNS response packet which exceeds 512 octets. Since DNSSEC protocol involves several additional records which are required for DNSSEC validation such as DNSKEY and RRSIG records, therefore it needs TCP fall-back process if querying client and DNS full resolver do not support EDNS0 (Extension mechanism for DNS) [12]. By enabling EDNS0, the size of DNS packet can be up to a maximum of 1024 - 4096 bytes which is possible for DNSSEC based name resolution without TCP fall-back process. Another typical issue is the increase of workload on DNS full resolvers. Since DNSSEC validation requires additional name resolutions, thus the increase of network traffic may oppress network bandwidth. This issue can be solved by tuning the performance of DNS full resolvers and introducing high speed network. In addition to above issues, we mainly focus on the following issues in this paper.

i When a DNSSEC enabled DNS full resolver fails on DNSSEC validation, it sends only “SERVFAIL” response to the querying client without any resource record. However, the querying client cannot identify the exact reason of the “SERVFAIL” response since there can be several situations causing the “SERV-
DNSSEC validation is performed on each querying client and a validation error message will alert the user when the DNSSEC validation fails or times out to make the user decide whether use the answer or not without successful DNSSEC validation.

3. Related Works

So far many reports and solutions related to DNSSEC have been released including potential challenges of migration from traditional DNS, deployment situation and load balancing, etc. In this paper, we only pick up and discuss several approaches [13], [14], [16] related to our purpose of solving the existing issues in the current DNSSEC protocol which were mentioned in Sect. 2.2.

Migault et al. [13] proposed a queried Fully Qualified Domain Name (FQDN) based load balancing architecture (instead of IP address based load balancing) to overcome DNSSEC performance issue. Furthermore, it also investigated an effective proactive caching method using Distributed Hash Table (DHT) protocol to optimize the resources of an Internet Service Provider (ISP) used on DNSSEC resolving operation. Migault et al. [14], proposed a FQDN based prefetching method to overcome DNSSEC performance issue. This approach optimizes the resources of name resolution platforms by practically using cache achieved from FQDN based prefetch mechanism. The purposes of these two works are to mitigate the workload increase and size extension of name resolution platform caused by the migration from traditional DNS to DNSSEC.

On the other hand, NLnet Labs [15] developed a tool named DNSSEC-trigger [16] which can perform DNSSEC validation on querying client when its corresponding official DNS full resolver does not support DNSSEC validation. The main purpose of DNSSEC-trigger is to complement DNSSEC feature for name resolution. Therefore, when there is one or more DNS full resolvers with DNSSEC feature enabled DNSSEC-trigger uses the DNS full resolvers for DNSSEC validation instead validating by itself.

In summary, the approaches to DNSSEC proposed so far were mainly for load balancing of DNS full resolvers and complement of DNSSEC validation feature in case there is no DNSSEC enabled DNS full resolver available. Therefore, these approaches cannot solve the issues we described in Sect. 2.2 fundamentally.

4. A Client Based DNSSEC Validation System with Adaptive Alert Mechanism

Based on the survey of existing issues in the current DNSSEC and related researches we propose a client based DNSSEC validation system with adaptive alert mechanism considering minimal client timeout. The main purpose of the proposed system is to solve the existing issues of the current DNSSEC protocol mentioned in Sect. 2.2. Therefore the contributions of the proposed system include covering the querying client with the benefit of DNSSEC protocol and providing DNSSEC validation even in an outside network without DNSSEC available DNS full resolver, and more importantly, alerting the user with a pop-up window in case of DNSSEC validation failure or timeout. Note that regarding the contribution of reducing the workload of DNS full resolvers, we have confirmed its effectiveness by similar evaluation in [17], therefore we omit the details in this paper. Note that, when “SERVFAIL” is returned from DNSSEC validation process, the querying client also receives corresponding resource records with alert pop-up window to inform the user of the DNSSEC validation failure or timeout. Furthermore, since “SERVFAIL” can be caused by DNSSEC validation failure or timeout, the proposed system also considers minimal client timeout in name resolutions. Consequently, the users are able to determine how to deal with the name resolution results with the alert messages.

4.1 Design

In the proposed system, we keep stub resolver of querying client as it is and add a proxy and inside DNS full resolver to accomplish name resolution with extra features. Basically, a querying client needs to perform DNSSEC validation by itself and also to indicate the user with alert messages in case of the DNSSEC validation failure or timeout. Since it is still in the design phase, considering the simplicity of the implementation, we designed a proxy for performing name resolution in querying client and also installed an independent inside DNS full resolver on each querying client for performing DNSSEC validation. Figure 2 shows a simple topology of the proposed system. As we can see from Fig. 2, the proxy listens on “127.0.0.1:53” and the inside DNS full resolver listens on “127.0.0.1:10053”. Accordingly, in the proposed system, name resolution request will be received by the proxy and performed by the proxy and inside DNS full resolver. We describe the brief name resolution proce-
forms normal name resolution without any alert message. If stub resolver requires DNSSEC validation and the normal name resolution ((a)-a) causes timeout, the proposed system just performs normal name resolution without sending back any alert message. Finally, if stub resolver requires DNSSEC validation and the timeout is caused by the DNSSEC validation, then the proposed system sends back the queried resource record without DNSSEC validation and with timeout alert message.

With the above mechanism, the proposed system not only can provide DNSSEC validation on each querying client but also indicates the user with the alert message in case of DNSSEC validation failure or timeout so that the user can determine the use of name resolution result practically. Consequently, the feature of sending alert message can effectively solve the issue (i) described in Sect. 2.2 by providing information about the validation failure while the feature of DNS full resolver added to the querying client can solve the issues (ii) and (iii). Note that in addition to the workload issue on DNS full resolver and DNSSEC validation on querying client, the proposed system also considers timeout caused by DNSSEC validation. Therefore, in that case, the propose system sends back name resolution result without DNSSEC validation and with timeout alert message before timeout occurs on the querying client. Based on the design we also implemented a prototype system and evaluated the feature in a local network environment as well as some typical DNS servers in the Internet. We describe the detail of implementation and evaluation in the following sections.

4.2 Implementation

We implemented a prototype system which consists of a proxy and an inside DNS full resolver. The proxy handles the whole name resolution process including query reception, response reply, DNSSEC validation and alert mechanism, etc. We implemented the proxy using The Perl Programming Language [18] and CPAN (The Comprehensive Perl Archive Network) [19] based on the design described in Sect. 4.1. Specifically, we achieved Net::DNS::SEC::Validator [20] perl module from CPAN which uses libval [5] (DNS name resolution library) and implemented the DNSSEC module by adding extra features to it. For the proxy, we customized Net::DNSServer::Proxy [21] and Net::DNSServer [22] modules which also can be achieved from CPAN. Finally, in order to implement the proposed system on a windows operating system, we used Cygwin terminal.

As we described in Sect. 4.1, as an additional important feature, in case of DNSSEC validation failure the proposed system displays alert message in a pop-up window. In order to accomplish this feature, we used a method named resolve_and_check() in the module Net::DNS::SEC::Validator
Table 1  DNSSEC validation status codes

<table>
<thead>
<tr>
<th>valStatus</th>
<th>valStatusStr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VAL_BOGUS</td>
</tr>
<tr>
<td>2</td>
<td>VAL_DNS_ERROR</td>
</tr>
<tr>
<td>4</td>
<td>VAL_NOTrust</td>
</tr>
<tr>
<td>128</td>
<td>VAL_SUCCESS</td>
</tr>
<tr>
<td>133</td>
<td>VAL_NONEXISTENT_NAME</td>
</tr>
<tr>
<td>134</td>
<td>VAL_NONEXISTENT_TYPE</td>
</tr>
<tr>
<td>135</td>
<td>VAL_NONEXISTENT_NAME_NOCHAIN</td>
</tr>
<tr>
<td>136</td>
<td>VAL_NONEXISTENT_TYPE_NOCHAIN</td>
</tr>
<tr>
<td>137</td>
<td>VAL_PINSECURE</td>
</tr>
<tr>
<td>138</td>
<td>VAL_PINSECURE_UNTRUSTED</td>
</tr>
<tr>
<td>139</td>
<td>VAL_BARE_RRSIG</td>
</tr>
<tr>
<td>140</td>
<td>VAL_IGNORE_VALIDATION</td>
</tr>
<tr>
<td>141</td>
<td>VAL_UNTRUSTED_ZONE</td>
</tr>
<tr>
<td>142</td>
<td>VAL_OOB_ANSWER</td>
</tr>
<tr>
<td>143</td>
<td>VAL_TRUSTED_ANSWER</td>
</tr>
<tr>
<td>144</td>
<td>VAL_VALIDATED_ANSWER</td>
</tr>
<tr>
<td>145</td>
<td>VAL_UNTRUSTED_ANSWER</td>
</tr>
</tbody>
</table>

and performed DNSSEC validation which returns back one of the status codes listed on the table Table 1 [23]. Based on the status values, the proposed system determines whether the DNSSEC validation is failed or not. From the table, the resolve_and_check() function returns VAL_SUCCESS (128) for successful DNSSEC validation of existing resource record, VAL_NONEXISTENT_NAME (133) for successful DNSSEC validation of non-existing domain name and VAL_NONEXISTENT_TYPE (134) for successful DNSSEC validation of non-existing resource record. On the contrary, when the DNSSEC validation for the DNS response fails or the verification of non-existence of trust train fails, then the status VAL_BOGUS will be returned. One more important feature is timeout alerting mechanism. We implemented this feature in the proxy and it measures the name resolution time taken by the inside DNS full resolver and the DNSSEC validation time taken by the DNSSEC module. We checked the default timeout in windows operating systems (Windows 7 and 10) using “nslookup” command and confirmed the value is 2 seconds. Therefore, considering the runtime of proxy and effective name resolution, we set the threshold of timeout in the proxy to 1.9 seconds.

Figure 3 presents the workflow of the proxy in the proposed system with comparison of conventional name resolution system. As shown in (a) of the figure, when the proxy receives a query from stub resolver, first, it checks if the query includes CD flag. If it does, the proxy forwards the query to the inside DNS full resolver and sends the corresponding name resolution results back to the stub resolver. On the other case, if the query does not include CD flag, the proxy forwards the query to the inside DNS full resolver with CD flag and meanwhile starts time measurement. If the proxy does not receive response from the inside DNS full resolver within 1.9 seconds, the proxy does not display timeout alert message and just sends back the corresponding name resolution results to the stub resolver. This may result in timeout but it is a normal name resolution process, therefore the proxy does not display the timeout alert message. On the other case, if the proxy receives response from the inside DNS full resolver within 1.9 seconds, the proxy stops the time measurement and starts measuring the rest time to 1.9 seconds. Then the proxy checks the status of the DNSSEC validation process. Differs from the proposed proxy, in the conventional name resolution system, as shown in (b) of the figure, stub resolver forwards all queries to the indicated DNS cache server in all cases and the DNS cache server only sends back “SERVFAIL” response when the DNSSEC validation is failed or timeout.

Figure 4 presents the workflow of checking DNSSEC validation process in the proxy. When the proxy receives a DNS response from the inside DNS full resolver within 1.9 seconds, first, it sets $vflag=0 and $tflag=0 which indicates “no DNSSEC validation failure” and “no timeout occurred”. Next, the process goes down to the following steps.

1. The proxy attracts the domain name, class and type from the question section from the DNS response and call resolve_and_check() function to perform DNSSEC validation. At the same time, the proxy goes into sleep status for (1.9-name resolution time taken in the inside DNS full resolver) seconds to wait for the results of the process which is running the resolve_and_check().

2. In the process of running resolve_and_check() function, the proxy checks the status value ($status) of the DNS response received from the resolve_and_check() function and takes different action properly. The available status values are listed in Table 1.

3. The $status of VAL_NOTrust(4) and VAL_BOGUS(1) mean DNSSEC validation failures. In these cases, the answer section of the DNS response received from the inside DNS full resolver will be deleted and the answer section of the DNS response received from the resolve_and_check() function will be used instead for the final DNS response. Then, the proxy will display an alert pop-up window indicating DNSSEC validation failure to the user.
4. The status of VAL_SUCCESS(128) or VAL_NONEXISTENT(133) or VAL_NONEXISTENT_TYPE(134) means the DNSSEC validation is successful. In these cases, similarly, the answer section of the DNS response received from the inside DNS full resolver will be deleted and the answer section received from the resolve_and_check() will be used instead and then the DNS response will be replied back to the stub resolver with AD (Authenticated Data) = 1.

5. If the status has other values than the above steps 3 and 4, then the received DNS response packet will be replied back to the stub resolver without any extra process.

6. On the other hand, the process including the above steps 2, 3 and 4 has time restriction. If the above steps 2, 3 and 4 are finished within (1.9–name resolution time taken in the inside DNS full resolver) seconds then the DNS response from the above step 3 and 4 will be replied to stub resolver without any timeout alerting ($tflag=0). Otherwise, only the answer receive from the inside DNS full resolver without DNSSEC validation will be replied to the stub resolver. In this case, an alert message indicating timeout caused by DNSSEC validation will be displayed to the user.

By realizing the workflow described in Fig. 3 and Fig. 4, we implemented a prototype of our proposed system. Note that in additional to the alert messages indicating DNSSEC validation failure, the proposed system also considers to minimize client timeout. As we can see from the workflow presented in Fig. 4, the proxy will sends the DNS response without DNSSEC validation back to the user with timeout alert message in case that the DNSSEC validation causes timeout. As a consequence, the user can practically use the name resolution results because of the adaptive alert mechanism.

5. Feature Evaluation and Results

We implemented the proxy using Perl 5.22.0 on a PC (Intel(R) Core(TM) i5-4210M CPU: 2.60GHz, RAM: 4GB, Operating System: Windows7 32bit) and installed BIND (Berkeley Internet Name Domain)[24] as well as Unbound [25] as the inside DNS full resolvers. In order to evaluate the features of the prototype system, we constructed a local experimental network as shown in Fig. 5 and performed evaluations using local experimental zones as well as global zones. Note that we only made the client connect to the Internet while the zone server was restricted to local communication only.

5.1 Feature Evaluations

In the local experimental zones, we configured private root zone, “com” zone and its sub zones such as “aaaa.com” and “aaaab.com”, etc. For DNSSEC configuration, we signed the private root zone, “com” zone and two of the sub zones named “aaaa.com” and “aaaab.com”. We also changed the DS (Delegation Signer) record of “aaaab.com” on “com” zone to reproduce the DNSSEC validation failure. For both ZSK (Zone Signing Key) and KSK (Key Signing Key) in the experimental zones, we used RSA/SHA-2 encryption algorithm. Finally, we added the public key of the private root zone as “trusted-keys” to the inside DNS full resolver to make it perform DNSSEC validation for the private zones. With these configurations on the zone server and inside DNS full resolver, we performed feature evaluations on the client using “dig” command. As a result, we confirmed the prototype system worked well as we expected including the cases.
Fig. 6 Comparison with conventional DNS full resolver

Then, we also performed feature evaluations on global zones. Here, we only describe the case of DNSSEC validation failure comparing with the result of Google Public DNS resolver [26] and omit the case of successful DNSSEC validation. Similarly, we ran “dig” command with “+dnssec” option using Google Public DNS which IP address is “8.8.8.8” and compared the results with the prototype system. In this case, we used the domain “dnssec-failed.org” which is used for broken DNSSEC validation test [27]. First, we ran “dig dnssec-failed.org +dnssec @8.8.8.8” on the prompt and the results are shown on (a) of Fig. 6. Since the Google Public DNS full resolver supports DNSSEC validation and the RRSIG of the domain name “dnssec-failed.org” was falsified intently, thus the DNS response only indicates “SERVFAIL” without any resource record. Consequently, the user cannot see any resource record when the DNSSEC validation failed in the conventional DNSSEC enabled name resolution. On the other hand, when we ran the same command using the prototype system, we can receive both the resource record and alert message as shown on (b) of Fig. 6.

The prototype system validates the DNSSEC signature by itself and sends back the validation result with the resource record to the user even when the DNSSEC validation is failed. Therefore, the user can determine whether trust the resource record or not considering the alert message and the corresponding reason. This is very useful when DNSSEC validation failure occurs due to some misconfiguration and communication troubles.

Next, we reproduced a test case of timeout caused by DNSSEC validation using a global zone named “jprs.jp”. When we started up the inside DNS full resolver, it has no cache information even if we have configured the hint information about root servers. Therefore, we could reproduce the timeout test case using the prototype system by querying the A record of “www.jprs.jp” with “+dnssec” option. We show the results on Fig. 7. Even the stub resolver could receive the answer, it also received two alert messages as shown in the figure. The alert message (a) in the figure indicates that the name resolution caused timeout due to the DNSSEC validation process took too long and the alert message (b) indicates that the DNS response is failed in DNSSEC validation. Therefore, the user can determine whether to user the DNS response or not.
Table 2  Latency of name resolution on prototype system
(Version information: BIND9.9.8-P2; Unbound1.5.9)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Inside BIND only (with cache)</th>
<th>Proxy + BIND (with cache)</th>
<th>Inside Unbound only (with cache)</th>
<th>Proxy + Unbound (with cache)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>4355 (1) ms</td>
<td>*1928 (84) ms</td>
<td>3961 (1) ms</td>
<td>2599 (84) ms</td>
</tr>
<tr>
<td>2nd</td>
<td>4971 (0) ms</td>
<td>*1935 (87) ms</td>
<td>3255 (0) ms</td>
<td>4269 (92) ms</td>
</tr>
<tr>
<td>3rd</td>
<td>2458 (1) ms</td>
<td>1140 (83) ms</td>
<td>3664 (0) ms</td>
<td>2371 (85) ms</td>
</tr>
<tr>
<td>4th</td>
<td>1050 (1) ms</td>
<td>*1927 (89) ms</td>
<td>1140 (0) ms</td>
<td>3959 (83) ms</td>
</tr>
<tr>
<td>5th</td>
<td>3518 (0) ms</td>
<td>*1927 (86) ms</td>
<td>3537 (0) ms</td>
<td>1147 (83) ms</td>
</tr>
<tr>
<td>6th</td>
<td>2823 (0) ms</td>
<td>964 (84) ms</td>
<td>2104 (0) ms</td>
<td>1961 (83) ms</td>
</tr>
<tr>
<td>7th</td>
<td>781 (1) ms</td>
<td>*2066 (87) ms</td>
<td>2501 (0) ms</td>
<td>4070 (84) ms</td>
</tr>
<tr>
<td>8th</td>
<td>3339 (0) ms</td>
<td>1138 (86) ms</td>
<td>2513 (1) ms</td>
<td>4858 (83) ms</td>
</tr>
<tr>
<td>9th</td>
<td>2689 (1) ms</td>
<td>*1928 (81) ms</td>
<td>3408 (0) ms</td>
<td>4821 (94) ms</td>
</tr>
<tr>
<td>10th</td>
<td>1549 (0) ms</td>
<td>1140 (91) ms</td>
<td>2507 (0) ms</td>
<td>3214 (85) ms</td>
</tr>
<tr>
<td>Avg</td>
<td>2753 (1) ms</td>
<td>1609 (86) ms</td>
<td>2879 (1) ms</td>
<td>3323 (86) ms</td>
</tr>
</tbody>
</table>

5.2 Latency Measurement

Considering that the prototype system may cause timeout due to the overhead of DNSSEC validation process. We measured the latency of name resolution on the prototype system using a global domain name “www.jprs.jp” by querying its A record with “+dnssec” option using “dig” command. We measured the latency in several patterns which include “inside DNS full resolver (BIND) only”, “proxy + inside DNS full resolver (BIND)”, “inside DNS full resolver (Unbound) only” and “proxy + inside DNS full resolver (Unbound)” and compared the results. Moreover, considering the case that the inside DNS full resolver has no cache information about the querying domain name, we performed the measurement twice continuously after running cache flush in the inside DNS full resolver and took ten samples for each pattern.

We present all the samples and the average latencies on Table 2. From the results, we can see that most name resolutions without using any cache took more than 2 seconds in all patterns even when the proxy is not used. This is because the inside DNS full resolver has no available cache information during the first time name resolution. On the other hand, the latencies of the second name resolution were reduced significantly in all patterns since the cache worked after the first name resolution was finished. Note that the pattern “Proxy+BIND” has low average latency because it includes 6 samples (marked with asterisk) timed out which were finished within 2 seconds without DNSSEC validation. The results are reasonable because we set a strict threshold (1.9 seconds) for timeout in the proxy and it can be improved by adjusting the threshold based on specific requirement.

Based on the above evaluation results, we can confirm that the prototype system can perform DNSSEC validation on querying client correctly and also can handle the received answers without DNSSEC validation based on the alert message as we expected. For the name resolution latency, though the first time of name resolution may cause timeout, the latency of the name resolution from the second time which can use the cache is also acceptable for real operation.

5.3 Simulation of Cache Poisoning Attack

In addition to the normal feature evaluation, in order to confirm that if the proposed system can correctly alert the DNSSEC validation failure caused by poisoning attack, we
simulated a cache poisoning attack by changing the DNS response in man-in-the-middle manner. It is difficult to simulate a cache poisoning attack in a real network environment, therefore we added a new proxy for cache poisoning between the inside DNS full resolver and the Internet as show in Fig. 8. The newly added proxy only change the DNS response of the query for the A record of the domain name “www.jprs.jp.” to a fake one. The simulation results are shown in Fig. 9. The part (a) shows the correct name resolution result while the part (b) shows the poisoned DNS response with the alert message. With the simulation we can confirm the proposed system can be applied in real network environment.

6. Discussion

Though the prototype system worked correctly as we designed and can solve the existing operational issues in the current DNSSEC, we consider that the system still needs some improvement for applying to real operation and contribution to the wide deployment of DNSSEC.

First, the proxy sends the same query to “DNSSEC module” to perform DNSSEC validation for every name resolution request with “+dnssec” option and it is not efficient. The procedure can be optimized by adding cache function to the proxy independently for those name resolutions results with successful and failed DNSSEC validation in addition to the inside DNS full resolver. Then, the name resolution for uncached domain name may results in timeout in the prototype system since the cached information will be gone every time the PC is restarted. We consider to solve this problem by proactively adding cache information into the inside DNS full resolver when it launches so that the latency for achieving well-known root zone and TLD (Top Level Domain) zones related information can be reduced. Next, what we need to mention is the capability of the proposed system on different operating system. In this paper, we only introduced a prototype on windows system but we also plan to scale it to other well-used operating system such as Mac OS and Linux systems. Moreover, we need to mention the performance of name resolution on querying client especially when it includes DNSSEC validation. We installed BIND and Unbound on multiple querying clients as inside DNS full resolvers and compared the performance of name resolution when they use their own DNS full resolver and use one official DNS full resolver on a local experimental network [17]. Through the comparison we confirmed that when there exist multiple querying clients their performance of name resolution in total could be higher than that of official DNS full resolver. Therefore we consider performance concern can be easily handled in our proposed system. Finally, we discuss the effect of client based DNSSEC validation on authoritative name server side. Since the name resolution results of the queries with CD flag will be cached in the local cache DNS server, therefore we consider that this increase of DNS queries on the authoritative name servers is not critical. As an alternative option, the querying client can use official DNS full resolver instead, but it will increase the workload of the official DNS full server. As a result, this problem is an tradeoff between official DNS full resolvers and authoritative name servers.

7. Conclusion

In this paper, we surveyed the current DNSSEC protocol as well as its existing operational issues, and based on the survey results we proposed a client based DNSSEC validation system with adaptive alert mechanism considering minimal client timeout to solve the existing operational issues. In the proposed system, in addition to client based DNSSEC validation, adaptive pop-up alert messages will be displayed to the user in case of DNSSEC validation failure and timeout. Moreover, the proposed system also considers minimal client timeout and replies name resolution results before the timeout caused by DNSSEC validation process occurs and displays a timeout alert message. Therefore, the user can determine whether trust the name resolution results without successful DNSSEC validation or not based on the alert messages. The proposed system can contribute to reduce workload of DNS full resolvers caused by DNSSEC validation and also can provide secure name resolution to querying client usually, but in the trouble cases the system also can send answers to the user instead just error messages. We also implemented a prototype of the proposed system and evaluated the features on a local experimental network and in the Internet. With the evaluation results, we confirmed that the prototype system worked well as we expected and it can contribute to promote DNSSEC deployment.

Future work includes performance tuning on real network environment and extension to other operating systems. After finish performance tuning, we also plan to introduce the proposed system into real clients for service providing, and at the same time, we plan to proceed the extension work.

Acknowledgments

This work was partially supported by JSPS KAKENHI (Grants-in-Aid for Scientific Research) Grant Number 15K16002.

References


Naoya Kitagawa received his B.Sc. and M.Sc. degree in information science from Chukyo University, Toyota, Japan in 2009 and 2011 respectively, and his Ph.D. degree in information science from Nagoya University, Nagoya, Japan in 2014. In April 2014, he joined Information Technology Center, Nagoya University as a postdoctoral fellow. Since October 2014, he has been an assistant professor in the Institute of Engineering, Tokyo University of Agriculture and Technology. His research interests include Internet, network security, and distributed system. He is a member of IEICE and IEEE.

Yong Jin received his M.E. degree in electronic and information systems engineering and Ph.D. degree in Industrial Innovation Sciences from Okayama University, Japan in 2009 and 2012 respectively. In April 2012, he joined the Network Architecture Laboratory of the Photonic Network Research Institute in the National Institute of Information and Communications Technology, Japan, as a researcher. From October 2013, he joined the Global Scientific Information and Computing Center of Tokyo Institute of Technology as an assistant professor. His research interests include network architecture, traffic engineering and Internet technology. He is a member of IPSJ and IEICE.

Kunitaka Kako received the B.S. degree in Information Technology from Tokyo University of Agriculture and Technology in 2016. He is currently a student in the Graduate School of Science in Information Technology, Tokyo University of Agriculture and Technology. His research interests include Computer network.

Nariyoshi Yamai received his B.E. and M.E. degrees in electronic engineering and his Ph.D. degree in information and computer science from Osaka University, Osaka, Japan, in 1984, 1986 and 1993, respectively. In April 1988, he joined the Department of Information Engineering, Nara National College of Technology, as a research associate. From April 1990 to March 1994, he was an Assistant Professor in the same department. In April 1994, he joined the Education Center for Information Processing, Osaka University, as a research associate. In April 1995, he joined the Computation Center, Osaka University, as an assistant professor. From November 1997 to March 2006, he joined the Computer Center, Okayama University, as an associate professor. From April 2006 to March 2014, he was a professor in the Information Technology Center (at present, the Center for Information Technology and Management), Okayama University. Since April 2014, he has been a professor in the Institute of Engineering, Tokyo University of Agriculture and Technology. His research interests include distributed system, network architecture and Internet. He is a member of IEICE and IEEE.
Masahiko Tomoishi received the B.E. and M.E. degrees in Computer Science from Tokyo Institute of Technology in 1989 and 1991, respectively. In 1994, he joined the Department of Computer Science, Tokyo Institute of Technology, as a research associate. In 2007, he joined National Institute of Informatics, as a specially appointed research associate. In 2008, he joined Tokyo Institute of Technology, as a specially appointed assistant professor. Since April 2011, he has been a professor in Tokyo Institute of Technology. His research interests include software verification and network/security management. He is a member of IEICE and IEEE.