A novel zone walking protection for secure DNS Server

***Abstract*—Zone walking attack is to get all existing domain information from a secured DNS server. NSEC3 protocol was proposed to defend against zone walking attack in a secured DNS server, although NSEC3 uses more CPU time. In this paper, we have proposed two novel solutions to defend against the zone walking attack by addressing the efficiency issue of secure DNS protocol. We have simulated our proposed solution and analyzed it with different scenarios of the secure DNS server and attackers. The result of our experiment shows that our proposed solution *Low Profiling* can be effective against zone walking attack for up to certain server-side and client-side parameters. Our work can help researchers to understand how a new approach in the DNSSEC server can defend against zone walking attack.**

***Index Terms*—DNS, DNSSEC, Zone walking, Low profiling**

I. Introduction

Initially, the Internet was designed only for limited users and with no security in mind. However, the number of Internet users has been increasing very rapidly. Many cyber attackers have come into the world of the Internet and focused on several security flaws in various networking protocols, thereby exploiting it.

DNS servers are used to look up the IP address against a name. Thus, the DNS server can provide the corresponding IP address from a domain name. However, DNS servers are not secured. A spoofed DNS server might provide a fake or malicious IP address for a requested domain name, thereby redirecting a client to some malicious web site. To address this problem, DNSSEC [1] has been proposed where the domain name and its IP address (along with HTTPS ability information) are mapped in a secured fashion. Currently, DNSSEC [1] is roughly above 80% implemented. DNSSEC has NSEC sub-protocol [2] to securely pass the non-existence of domain names. Nevertheless, the problem is that NSEC is vulnerable to zone walking attack.

Zone walking attack is a kind of privacy invasion into the DNS records. The zone walking attack is to get all existing domain information from the DNSSEC server. The fetched information might contain some domain names and their detailed information. To prevent zone walking attack, NSEC3 [2] protocol has been proposed and implemented in DNSSEC. But NSEC3 [2] is comparatively slower than NSEC.

The existing solution NSEC3 needs domain name hashing in DNSSEC serversNSEC3 [2]. Using NSEC3, the zone walking attack is not possible, because an intruder cannot do reverse engineering domain names from the received hashed values. However, since domain name hashing is mandatory for NSEC3, it uses more CPU time in DNSSEC servers.

In the paper, we have addressed the efficiency issue in NSEC3 in the existing DNSSEC servers to defend against zone walking attack. To the best of our knowledge, there exists no previous work that attempts to deals with the performance improvement of the NSEC3 approach. This is the *novelty* of our approach.

Our proposed mechanism can be utilized in DNSSEC servers to prevent zone walking attack. Our solution *Low Profiling* can be a better alternative to NSEC3 in DNSSEC servers. Because it does not use any domain name hashing, rather it profiles clients to detect suspicious clients.

To analyze our proposed solution, we have simulated a simplified DNSSEC server using Java programming language. The zone walking attack was also simulated [3]. Then, we have changed both server-side (e.g., *Server Tolerance*) and client-side (e.g., *Attack Noise*) parameters to analyze different scenarios of the DNSSEC server and attackers. Then, we have carefully plotted (i.e., considering experimental error) several graphs for these scenarios.

The result of our experiment shows that our solution *Low Profiling* can be effective against zone walking attack for up to specific server-side and client-side parameters. Therefore, we have realized the usefulness of our proposed solution against the zone walking attack.

Our work can help researchers to understand how a new approach in the DNSSEC server can defend against the zone walking attack. It can also be useful to understand how the different approaches of the DNSSEC server (i.e., server parameters) can play against zone walking attacks of different strengths (i.e., *Attack Noise*), and to understand how effective DNSSEC server's approach is. In the future, a variant of our proposed defensive approach can be built upon the experiment's result.

The rest of the paper is organized as follows. In Section II, we have described related terminologies to the DNS system. In Section III, we have described our two proposed approaches: dividing list and Low Profiling. In Section IV, we have described the algorithm and the simulation setup of *Low Profiling* experiment. In Section V, we have analyzed the results of our experiment. Finally, Section VI has the concluding remarks.

II. Background and related works

A. DNS and DNSSEC

The basic infrastructure of DNS and DNSSEC [1] servers is the same. DNSSEC includes additional security measure to securely send the information of a domain name. DNSSEC is a security extension to the DNS server. DNSSEC [1] is defined as domain name system security extensions. Fig. 1 shows how DNS or DNSSEC works.

1) Client types www.example.com in the search bar of a web browser and this query are passed to recursive DNS server to fetch the IP address of this domain name.

2) Recursive DNS server sends an iterative query to root DNS server for the IP address of this domain name.
3) Root DNS server
   - sends a referral to the authoritative server for the com zone.
   - RRset of DNSkey records for the root zone, i.e. public ZSK and KSK of root zone.
   - RRsig of DNSkey records signed using private KSK of the root zone.
   - DS record for com zone, i.e. fingerprints of public KSK of com zone.
   - RRsig of DS record signed using private ZSK of the root zone.

   Recursive DNS server corroborates the root zone using public KSK. It decrypts RRsig of RRset using public KSK of the root zone. It also decrypts RRsig of DS record using public ZSK of the root zone.
4) Recursive DNS server sends an iterative query to the authoritative DNS server for com zone for the IP address of www.example.com.
5) com DNS server
   - sends a referral to the authoritative server for the example zone.
   - RRset of DNSkey records for the com zone, i.e., public ZSK and KSK of com zone.
   - RRsig of DNSkey records signed using private KSK of com zone.
   - DS record for the example zone, i.e., fingerprints of public KSK of example zone.
   - RRsig of DS record signed using private ZSK of com zone.

   Recursive DNS server corroborates com zone comparing the fingerprint of public KSK of com zone with DS record for com zone, which was obtained from the root zone. It decrypts RRsig of RRset using public KSK of com zone. It also decrypts RRsig of DS record using public ZSK of com zone.
6) Recursive DNS server sends an iterative query to the authoritative DNS server for the example zone for the IP address of www.example.com.
7) example DNS server
   - RRset of DNSkey records for the example zone, i.e., public ZSK and KSK of example zone.
   - RRsig of DNSkey records signed using private KSK of example zone.
   - RRset of A record from example zone, i.e., fingerprints of public KSK of example zone.
   - RRsig of A record signed using private ZSK of com zone.

   Recursive DNS server corroborates example zone comparing the fingerprint of public KSK of example zone with DS record for the example zone, which was obtained from com zone. It decrypts RRsig of RRset using public KSK of example zone. It also decrypts RRsig of A record using public ZSK of example zone.
8) Recursive DNS server sends the IP address of www.example.com.

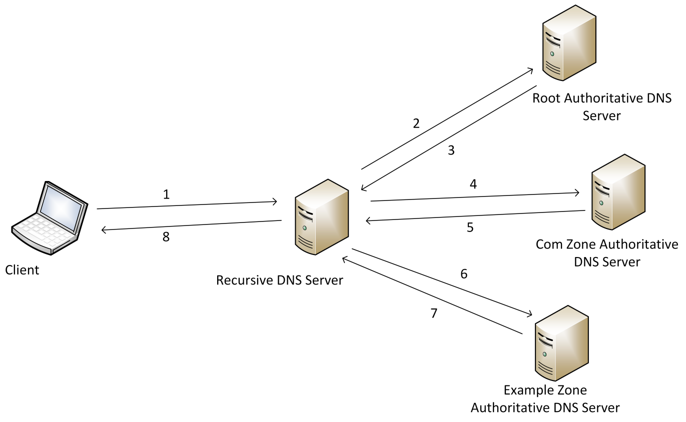

Root Authoritative DNS Server
2
3
1
8
4
5
Client
Recursive DNS Server
Com Zone Authoritative DNS Server
6
7
Example Zone Authoritative DNS Server

Figure 1: DNS or DNSSEC process

B. NSEC

DNS queries have two fields: name and type. Resource record (RR) is a response to a query. Each RR has four fields: name, value, type, TTL (time to live). The name and value field depends on the type of query and response. DNSSEC uses public-key cryptography to sign and authenticate DNS resource record sets (RR sets). Digital signatures are stored in RRsig resource records and are used in the DNSSEC [1] authentication process. An RRsig record contains the signature for an RRset with a particular name, class, and type. NSEC (Next Secure) is a RR that links to the next record name in the zone (in the canonical ordering of the zone) and lists the record types that exist for the record's name. These records can be used by resolvers to verify the non-existence of a record name and type as part of DNSSEC validation. [4] NSEC-records have the following data elements:

- Next domain name: The next record name in the zone (in the canonical ordering).
- Record types: The DNS record types that exist for the name of this NSEC-record.

Each NSEC record [2] is signed by the RRsig Resource Record so that it can not be corrupted. The next domain name contains authoritative data. The complete set of NSEC RRs in a zone indicates which authoritative RRsets exist in a zone and also form a chain of authoritative owner names in the zone. This information is used to provide authenticated denial of existence for DNS data.

The following NSEC RR identifies the RRsets associated with a.example.com. and identifies the next authoritative name after a.example.com.

a.example.com. 86400 IN NSEC b.example.com. (A MX RRSIG NSEC)

The first four text fields specify the name, TTL, Class, and RR type (NSEC). The entry b.example.com is the next authoritative name after a.example.com in canonical order. The A, MX, RRsig and NSEC mnemonics indicate that there are A, MX, RRSIG and NSEC RRsets associated with the name a.example.com. Assuming that the validator can authenticate this NSEC record, it could be used to prove that x.example.com does not exist, or to prove that there is no AAAA record associated with a.example.com.

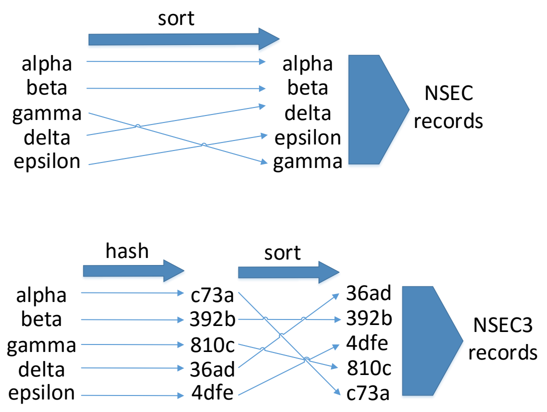
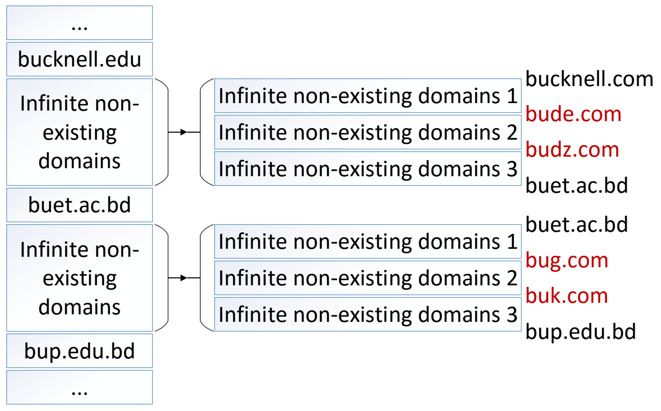


Figure 2: NSEC vs NSEC3



...
bucknell.edu
Infinite non-existing domains
buet.ac.bd
Infinite non-existing domains
bup.edu.bd
...
Infinite non-existing domains 1
Infinite non-existing domains 2
Infinite non-existing domains 3
Infinite non-existing domains 1
Infinite non-existing domains 2
Infinite non-existing domains 3
bucknell.com
bude.com
budz.com
buet.ac.bd
buet.ac.bd
bug.com
buk.com
bup.edu.bd

Figure 3: Dividing List mechanism

The main problem of NSEC [2] is that it allows enumeration of zone contents. An attacker can successively go through the NSEC chain and can thus determine all the entries of a zone, which is known as zone walking.

C. *Zone Walking Attack*

Zone walking might be seen by some domain administrators as a privacy invasion. Zone walking attack is to get all existing domain information from the DNSSEC server. The fetched information might contain some domain names and its detailed information, such as its mail server's IP address, which might not be intended by domain administrators.

1) Attack procedure: After the introduction of NSEC, it is easy to prove whether a domain name exists or not. If a domain name does not exist, then the range of non-existing domain names are sent to the requested client with a valid certificate. However, the problem is that an attacker can easily determine the next valid domain name after the non-existing domain. An attacker can easily chain through all non-existing domain ranges to find all valid domain names. Then the attacker can easily fetch all the data of the valid domain names.

D. *NSEC3*

NSEC3 [2] has been created to mitigate 'zone walking' which is possible with NSEC. NSEC3 does so by hashing domain names using SHA2 [2]. Fig. 2 shows where the hashing is used [5]. From the figure, it is clear that 'zone walking' is not possible since an intruder cannot do reverse engineering domain names from the received hashed values.

III. Proposed Approach

In this section, we explain our two proposed approaches as protection against zone walking attack: i) dividing list, and ii) low profiling.

A. *Dividing List*

The idea is actually based on NSEC, but it significantly slows down the zone walking attack. In this technique, instead of proving the next record name in the zone, another non-existing name is provided. For example, suppose all the names are mapped to a number for a better explanation. Let '1' be a valid name. The next valid name is '100'. If a client requests for some number in (1, 100) range, then non-existing (1, 100) range is provided by the rule of NSEC. However, instead, if (1, 10) range was provided, then zone walking would be slower by ten times. This idea has a trade-off. That is - instead of creating one certificate for (1, 100) range, ten certificates have to be created. The number (here 10) can be tuned so that the technique can mitigate zone walking with a minimal performance impact. Fig. 3 illustrates our proposed Dividing List mechanism.

1) Cost-Benefit Analysis:

- Cost: Since in this technique, the continuous range is divided into some value k, the time complexity of the technique depends on choosing k. The more value of k, the more time complexity but, the less likely to occur zone walking.
- Benefit: It has already been stated that zone walking will be slower by the technique.
- Limitation: It does not prevent zone walking attack; rather, it only slows down the attack. Also, it uses more CPU time in the order of k (already stated).

B. *Low Profiling*

In this technique, client requests are profiled to identify zone walking attackers. Zone walking has a pattern in client requests. Zone walking is done in alphabetical order. If the DNSSEC server can detect the pattern, then zone walking attackers can be easily identified. To detect the pattern, low profiling can be done. It has a minimal performance impact. Because we are only interested here in specific requests. For example, suppose all the names are mapped to a number for a better explanation. Let '1' is a valid name. The next valid name is '100'. If a client requests for some number in (1, 100) range, then non-existing (1, 100) range is provided by the rule of NSEC. So the attacker will then request for '100' to get its RRset. Then the attacker will request something greater than

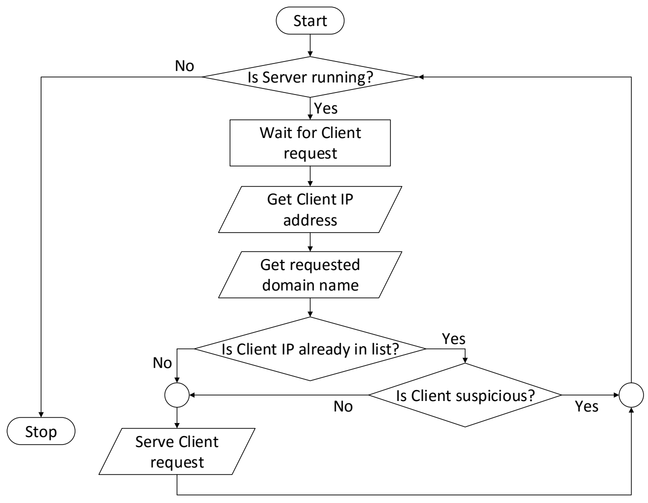


Figure 4: Low Profiling flowchart

'100'. By low profiling, the DNSSEC server can block the attacker. It can be done by checking the last domain requests of the same client is in lexicographical order, but the technique is ineffective if the attacker changes IP address in the middle of the zone walking attack. The good news is, the attacker has to change its IP address to bypass the low profiling technique, which is relatively harder to do from the perspective of the attacker. DNSSEC server can intelligently log client-requested data to identify attacker without using too much memory.

Pseudocode 1 shows the basic algorithm of this approach. Fig. 4 shows the flowchart of the Low Profiling mechanism.

IV. Implementation

A. Algorithm

Algo. 1 shows the algorithm for a low profiling mechanism. Here, $activityIsOk(ip)$ checks if the last domain requests of the client are all in lexicographical order, thereby sensing a zone walking attack by the requesting client. If such a suspicious client is detected (i.e., the domain requests are all in lexicographical order), the client is blocked from the DNSSEC server for some time.

B. Cost-Benefit Analysis

- Cost: Since, in the technique, a short-time client request is stored for background analysis, it uses some memory. When a specific client only requests for domain IPs with some non-existing domains by alphabetical order, the client will be rejected by the DNSSEC server temporarily to prevent possible zone walking attacks.
- Benefit: The technique is better than the technique described in Section III-A. Because the detection of attackers is more accurate than the technique described in Section III-A. Also, the technique prevents zone walking attack.
- Limitation: If the attacker is intelligent enough to change the client IP address in each request, or if the attacker owns multiple client IPs, then this low-profiling technique cannot prevent it.

Algorithm 1 Low profiling

```
$clientIpList \leftarrow \{\}$
$activityList \leftarrow \{\}$
**while** $isServerRunning$ **do**
    $waitForRequest()$
    $ip \leftarrow getClientIp()$
    $req \leftarrow getRequestedDomain()$
    $time \leftarrow getTime()$
    **if** $ip$ is in $clientIpList$ **then**
        $updateTime(ip, time)$
        $addToActivity(ip, req)$
        **if** $activityIsOk(ip)$ **then**
            $serveReq(ip, req)$
        **end if**
    **else**
        $addToClient(ip)$
        $updateTime(ip, time)$
        $addToActivity(req)$
        $serveReq(ip, req)$
    **end if**
    $cleanUpActivityList()$
**end while**
```

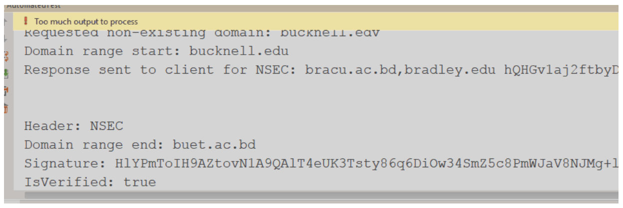
C. Simulation setup

A simulation has been self-created using Java language [3]. The simulation project has been created in an appropriate design pattern so that any DNSSEC mechanism and attacking mechanism can be plugged into the project. Here are the setup details:

1) DNSSEC Server: It is multi-threaded server. It has one thread per client. The number of maximum clients to the server has been deliberately limited to 10. It has been done to simulate real-life congestion of the DNSSEC server. For simplification, the DNSSEC server has been seen as a single server instead of multiple servers. All server activities are shown in console while running the simulation. 5 Any new server protection mechanism can be plugged into the project without changing the current code.

2) Client: Each client has one thread, whether it is a legitimate client or attacker. The client parses server response intelligently. It can detect if the server blocks the client because of suspicious activities. Any new client request mechanism can be plugged into the project without changing the current code.

3) Running test: To test an attacker mechanism, all the attackers have been run once. Since the simulation server can serve only ten clients, all other clients are queued. When one or more out of ten clients are served, the waiting clients are served by dequeuing clients one by one. The attacker test runs quickly because ten clients are served at a time. However, the server test runs slower because only one server is active at a time. This has been done to prevent opening a significant amount of ports in the simulation computer.


Too much output to process

```
Requested non-existing domain: bucknell.edv
Domain range start: bucknell.edu
Response sent to client for NSEC: bracu.ac.bd,bradley.edu hQHGv1aj2ftbyD


Header: NSEC
Domain range end: buet.ac.bd
Signature: HlYPmToIH9AZtovN1A9QAlT4eUK3Tsty86q6DiOw34SmZ5c8PmWJaV8NJMg+l
IsVerified: true
```

Figure 5: Simulation console screenshot

D. Parameter description

The results contain the data of one input column for attacker (*Attack Noise*) and another input column for server (*Server Tolerance*):

- Attack Noise (for an attacker): In zone walking attack, attacker searches domain as alphabetical order. However, the server can easily detect this type of behavior. So, the attacker needs to be tricky and break the serial attack. Attack Noise is the probability of breaking alphabetical order of domain query to server. Attack Noise 0.0 means no serial breaking, and 1.0 means every query is out of serial. The amount of noise on a scale of 0.0 to 1.0 (inclusive), higher noise for a stronger attack.
- Server Tolerance (for server): *Server Tolerance* (the number of suspicious records) is the number of continuous requests received alphabetically from a client needed by DNSSEC server to identify the client as an attacker. The lesser Server Tolerance, the more secure the server is. The higher Server Tolerance means the server will provide a response for more requests from an attacker.

Each input column has four output columns common for both attacker and server:

- *Attack Coverage*: The ratio between the number of domains fetched by the attacker and the number of domains stored in the server
- *Attack Runtime (msec)*: The elapsed runtime of the attacker client (in milliseconds)
- *Attack Speed (domain per msec)*: The speed of fetching domain by the attacker (in the number of domains fetched per millisecond)

E. Plotting Graphs

We have noted the data of the parameters described in Subsection IV-D from our experiment. To eliminate experimental error, we have used discrete X values. For a single X value, we have taken the average of the Y values of the corresponding range of X values. To better understand, here is an example. Say, we have selected X values such as: 0.2, 0.3, 0.4, and up to 1.0 (incriminating X values by 0.1). To plot the Y value of 0.2 X value, we have calculated the average of the Y values of the range of X values from 0.15 until 0.25, i.e., [0.15, 0.25). Similarly, to plot the Y value of 0.3 X value, we have calculated the average of the Y values of the X value range of [0.25, 0.35). So, if a Y value of a certain X value is erroneous, it does not affect the plotted graph too much. Thus, we have reduced the possible experimental errors.

V. Results

A. Attacker Side Results

All the graphs are drawn for *Server Tolerance* (e.g., the number of suspicious records) is 10. That means, after ten subsequent suspicious requests from a client, the DNSSEC server will identify the client as an attacker and block it.

1) Attack Noise vs. Attack Coverage: In Fig 6, we can see that *Attack Coverage* is low for low attack noise and high for high attack noise. Because, if the probability of breaking alphabetical order of requested domains by an attacker is low, then the DNSSEC server can easily identify that attacker after ten subsequent requests (as *Server Tolerance* is 10). So, in that case, the number of domains fetched from the server by the attacker is low. Similarly, if the probability of breaking the alphabetical order of requested domains by an attacker is high, then the server cannot easily identify that attacker. So, in that case, the domain fetched from the server by the attacker is high. Therefore, high attack noise means a stronger attacker, and low attack noise means a weaker attacker.

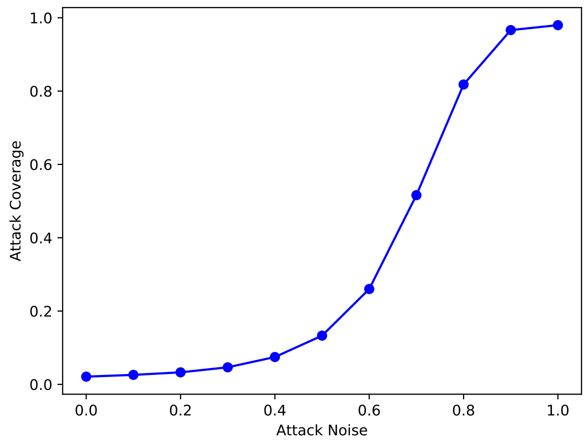


Figure 6: Attack Noise vs. Attack Coverage

2) Attack Noise vs. Attack Runtime (msec): The runtime of attack increases almost linearly for every noise level of attack. As attack noise increases, the attacker will be able to retrieve more domains from the server, as shown in Fig 7. In other words, the DNSSEC server needs more time to identify a client as an attacker if the amount of attack noise is higher.

3) Attack Noise vs. Attack Speed (domain per msec): Domain retrieved per millisecond increases slowly (in Fig 8) as noise increases for attacker because the rate of domain received from server is more than the runtime (from Fig 6 and Fig 7).

B. Server Side Results

All of the graphs of the server-side is based on attack noise 0.5.

1) Server Tolerance vs. Attack Coverage: In Fig 9, we can see that as *Server Tolerance* (i.e., the number of suspicious records needed to identify an attacker) increases, the domains

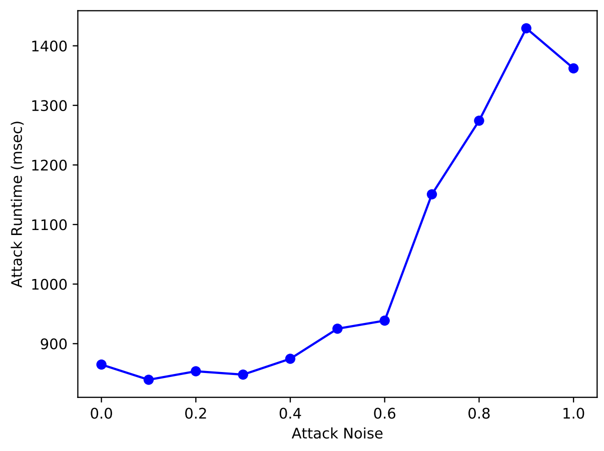


Figure 7: Attack Noise vs. Attack Runtime (msec)

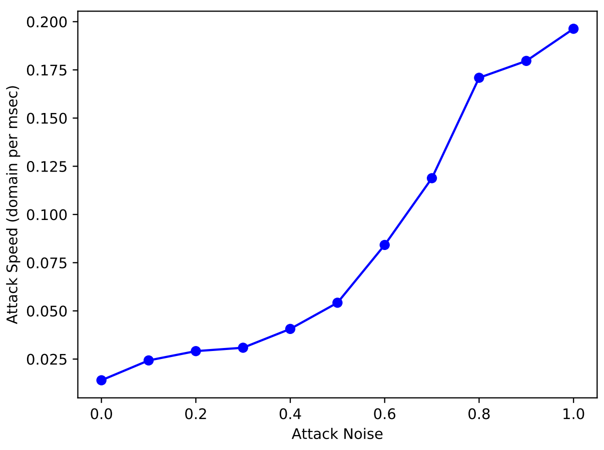


Figure 8: Attack Noise vs. Attack Speed (domain per msec)

fetched by an attacker from DNSSEC server increases proportionately up to some total suspicious records. This threshold point of Server Tolerance depends on the value of attack noise. After that threshold record, Attack Coverage remains fixed. Because, after this threshold point, the server cannot detect the requester (client) as an attacker anymore.

2) Server Tolerance vs. Attack Runtime (msec): In Fig 10, we can see that, initially, as Server Tolerance increases, the attacker runtime will increase proportionally. After a specific suspicious record, the attacker will get all the records from the DNSSEC server, so the runtime to receive these records will be high and fixed approximately.

3) Server Tolerance vs Attack Speed (domain per msec): In Fig 11, we can see; initially, attack speed (domains per msec) slowly increases because of the increase of the rate of domain received is more than the increase of the runtime of attack. After a certain number of suspicious records, the attack speed will remain constant approximately.

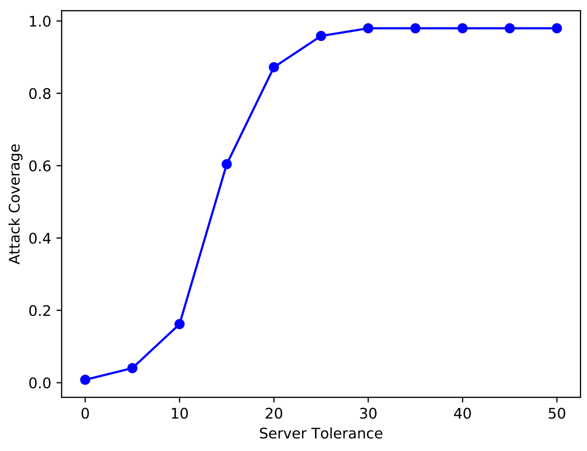


Figure 9: Server Tolerance vs. Attack Coverage

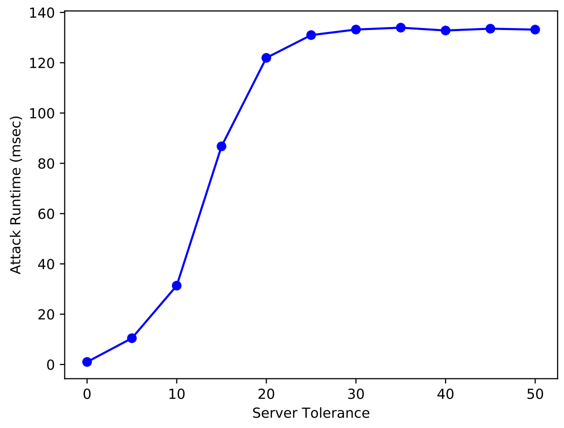


Figure 10: Server Tolerance vs. Attack Runtime (msec)

C. Server-Client Results

1) Server Tolerance vs. Attack Coverage with Attack Noise: In Fig 12, we can see how attack noise can affect attack coverage for different Server Tolerance. As stated earlier, higher attack noise means a stronger attacker who can fetch more domains even with limited server tolerance. Similarly, a weak attack (with attack noise of 0.1) cannot fetch all the domains even with a high value of server tolerance.

D. Discussion

In our experiment, we have used a simplified DNSSEC server to analyze our defensive idea against zone walking attack. However, the real world DNSSEC server is not that simple like our simulated DNSSEC server. For example, in the real world, the DNSSEC server exists in a hierarchy (as shown in Fig. 1), not as a single entity. Thus, the measurement of attack runtime in our experiment may be slightly different from the real world measurement. However, we believe that the real-world data is not so much different from the experimented data that can rebut the experimental result.

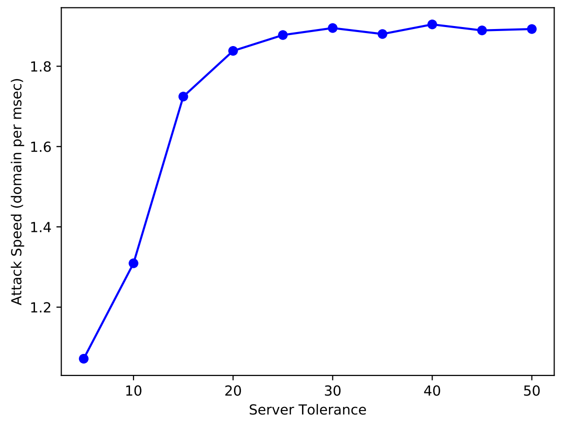


Figure 11: Server Tolerance vs. Attack Speed (domain per msec)

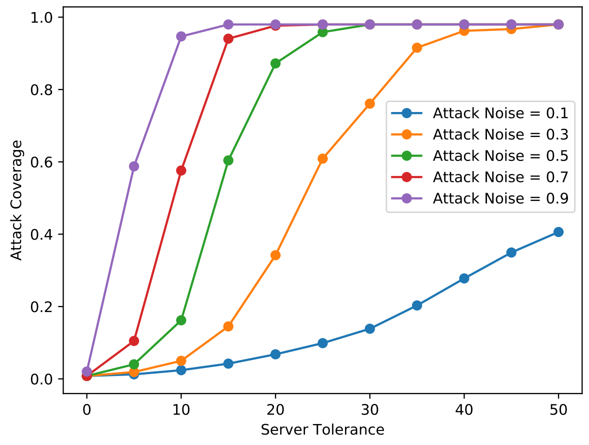


Figure 12: Server Tolerance vs. Attack Coverage with Attack Noise

VI. Conclusion

Zone walking attack attempts to get all existing domain information from a secured DNS server. Although the NSEC3 protocol was proposed to defend against zone walking attack, it takes much time to protect against such an attack. In this paper, we have proposed two novel defense mechanisms against zone walking attack to mitigate the intensity of such an attack. We have implemented those two approaches and tested them in the simulation environment. We have presented our results for different performance metrics. Our experiment can help researchers to understand how DNSSEC server's different approach (i.e., server parameters) can play against zone walking attacks of different strengths (i.e., attack noise), and to understand how effective DNSSEC server's approach is.

References

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," *IETF RFC 4033*, March 2005. [Online]. Available: https://tools.ietf.org/html/rfc4033

[2] S. Weiler and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence," *IETF RFC 6840*, February 2013. [Online]. Available: https://tools.ietf.org/html/rfc6840

[3] A. Paul, "DNSSEC Simulation," 2018. [Online]. Available: https://github.com/arnobpl/DNSSEC

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource Records for the DNS Security Extensions," *IETF RFC 4034*, March 2005. [Online]. Available: https://tools.ietf.org/html/rfc4034

[5] Niobos, "DNSSEC – the NSEC and NSEC3 record," January 20, 2010. [Online]. Available: https://blog.dest-unreach.be/2010/01/20/dnssec-the-nsec-and-nsec3-record