A Light-weighted Data Collection Method for DNS Simulation on the Cyber Range

Shuang Li[1], Shasha Du[1], Wenfeng Huang[1], Siyu Liang[1], Jinxi Deng[1], Le Wang[1], Huiwu Huang[2*], Xinhai Liao[3*], and Shen Su[1]
[1] Guangzhou University, Guangzhou, China
[2]Guangdong University of Technology, Guangzhou, China
[3]Guangdong University of Foreign Studies, Guangzhou, China
*Corresponding author: Huiwu Huang (393575614@qq.com), and Xinhai Liao (lxh@gdufs.edu.cn)

Received March 19, 2020; revised June 2, 2020; accepted July 11, 2020; published August 31, 2020

Abstract

The method of DNS data collection is one of the most important parts of DNS simulation. DNS data contains a lot of information. When it comes to analyzing the DNS security issues by simulation on the cyber range with customized features, we only need some of them, such as IP address, domain name information, etc. Therefore, the data we need are supposed to be light-weighted and easy to manipulate. Many researchers have designed different schemes to obtain their datasets, such as LDplayer and Thales system. However, existing solutions consume excessive computational resources, which are not necessary for DNS security simulation. In this paper, we propose a light-weighted active data collection method to prepare the datasets for DNS simulation on cyber range. We evaluate the performance of the method and prove that it can collect DNS data in a short time and store the collected data at a lower storage cost. In addition, we give two examples to illustrate how our method can be used in a variety of applications.

Keywords: DNS Data Collection, DNS Measurement, DNS Simulation, Cyber Range, Active Data Collection

http://doi.org/10.3837/tiis.2020.08.020 **ISSN : 1976-7277**

1. Introduction

The Domain Name System (DNS) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network [1], [2]. However, the current DNS root resolution architecture has the risk of power abuse which posts threats on the openness and equality of the Internet, for example, a top-level domain may be deleted from the root zone [3], [4].

To investigate and improve the DNS architecture, a cyber range can be used to simulate and test its security issues [5], [6]. In DNS simulation, DNS data collection is very important. The data required for DNS simulation varies for different experimental purposes. The data required for such scenarios must be accurate and light-weighted, which means the amount of data is not very huge and its content is organized.

Many researchers have made great efforts in DNS data collection and measurement. For example, LDplayer [7], Thales system [8], etc., all provided their solutions for collecting DNS data. However, the existing DNS data collection methods we found are not suitable for the needs of some scenarios due to the redundancy and large amount of the collected data. Excessive DNS requests require an enormous amount of computational resource. Generally, in order to collect various types of data in a limited time, the query traffic generated by most data collection systems is also huge. It not only brings much pressure on the requested server but also affects the normal DNS resolution.

For DNS security simulation, a zone file containing the domain name and IP address is very important, but the rest of DNS data is unnecessary, such as MX record and SRV record. The use of complete DNS data for security simulation may have the following problems: first, collecting and processing DNS data may consume excessive computing resources; secondly, the accuracy of the collected data cannot be not guaranteed, which will directly affect the accuracy of the zone file in the final generation.

Our idea is to provide a flexible way to simply and efficiently collect data for DNS measurements. In the process of data collection, we filter and clean up the data, as well as add relevant attributes according to the needs of the actual application so that the generated zone file can be more suitable for practical application. In this paper, we have designed a complete set of DNS data processing modules, and provide a suitable data set for DNS security simulation.

The rest of this paper is organized as follows. In Section 2, we introduce the related works of data collection for DNS measurements. In Section 3, we make the statement of our framework and the corresponding detailed introduction is in Section 4, 5, 6. We then give two examples of our system in practical applications in Section 7 and evaluate our data collection method in Section 8. Finally, we conclude this paper in Section 9.

2. Related Work

The measurement and analysis of Internet, especially DNS, provides a technique platform for improving network management, increasing network availability and avoiding large-scale network attacks [9]–[11]. According to the measurement method, it can be divided into the passive measurement and the active measurement.

2.1 The passive measurement

In the passive measurement method, the probes that record network activity are connected to the network, and the connections between network nodes are probed to summarize and record information about business traffic on that connection. Refs. [12] was the first to propose the collection of passive DNS data for network measurement in 2004. Refs. [13] discussed how to use passive DNS data to discover security events using domain names in 2007. Refs. [14] introduced a scalable method to manage the growing passive DNS data collection while correlating zone and network attributes in 2008. Refs. [15] described and built a disposable zone miner to automatically find disposable domain names in 2014. Their passive DNS data collection systems captured the above and below traffic of the recursive DNS server related to a large ISP in the Midwestern US. Refs. [7] proposed LDplayer, a DNS traffic replay tracker used for DNS evaluation in 2018. In the data collection part, LDplayer used the existing domain name list and a cacheless recursive server as the medium. Then it captured the entire records when accessing the domain name using packet capture, including the process of iterative access to root, TLD and STLD.

Passive measurement mainly observes the behavior of the network at a special point, and does not increase or modify the data load passing through the network, so it has no effect on the behavior of the network. This method can achieve a thorough understanding of the behavior of the network at the observation point, but it is difficult to obtain the overall understanding of the end-to-end behavior of the network [9], [16].

2.2 The active measurement

In the active measurement method, the behavior of the network is studied by sending data to the network, observing the results and the time required to send the data. In 2015, refs. [17] introduced Censys, a public query engine and data processing facility backed by data collected from ongoing Internet-wide scans. It provided researchers with the ability to scan the entire IPv4 space and use the results for open security research. Refs. [8] proposed the Thales system in 2016, which actively queried and collected records for massive amounts of domain names from various seeds. It used a seed for generating a list of domains, a LXC Farm for generating plenty of DNS requests, several servers for collecting data. Censys was not designed to scan the domain name space, rather, IPv4. Thales was designed to deal with DNS scanning and also complemented Censys.

Because the measurement is active and the collection range is controllable, we can consider the global range or the local range. The benefit of this is that the range can be adjusted according to demand, which is more flexible than passive collection. However, the active measurement method collects information through the request and response mode, so the network load will be caused during the collection process, and the data quality will be affected by network environment factors.

The measurement methods mentioned above suggest some ways for collecting massive real records of DNS. However, their collecting methods use plenty of computational resource and they can not work well within the limited resources. For example, LDplayer generated plenty of real-world data with around 60GB per day in February and around 145GB per day in December, and the total size of data is 2.67TB in their full passive dataset. The Thales system consists of two parts: traffic generator and data collector, including 64 processing cores and 164GB of RAM in each of two main parts.

In this paper, we provide a flexible way to simply and efficiently collect data for DNS measurements. Our data collection method has the advantages of the active measurement

method, and takes into account the device limitations and resource consumption of the data collection. Furthermore, the collected data can be used in plenty of experiments such as behavior tracking [17]–[20], malicious payload detecting [21], DNS security vulnerabilities mining [22], [23], classification, etc [24], [25].

3. Architecture: Overview

3.1 The Requirements of DNS on the Cyber Range

A Cyber Range (CR) offers an environment to implement a variety of simulation experiments, such as penetration testing, hardening critical infrastructure and testing with cyber security products [10], [26]–[28]. It provides isolation from other networks to deter malicious activity and represents complex networks in the real world [29], [30].

Our goal is to implement a light-weighted data collection method for DNS simulation on CR. In other words, we plan to apply the data to build a DNS simulation system on the cyber range, and CR here is just a kind of network scenario. We should implement the data collection method for DNS. The Data is the output of the method, that is the core of evaluating whether the collection method is lightweight. For the sake of analysis, we formally describe this requirement, which can be defined as a 5-tuple (R,t,a,r,c). The root zone R, the top-level domain zone t, the else authoritative zone a, recursive server r and client c. We also define three common functions to calculate the number of server nodes $N_s(\cdot)$, the number of zone files $N_z(\cdot)$ and the storage cost $S(\cdot)$.

The CR is very powerful, nevertheless, it's RAM and storage have limited capabilities for completely constructing the immense DNS system, which is hindered by funds. To solve this problem, we propose a light-weighted data collection method, which is briefly explained in Section 3.2. The processed data generated by this method is used to build a DNS system simulation application on CR, which will be described in detail in Section 7.

3.2 System Architecture

In Section 2, we compared the characteristics of the passive method with the active method. The active method collects a finer and larger range of data. In the context of limited resources, e.g. the cyber range, it is obvious that this method is more suitable for minimizing the collection of comprehensive and effective DNS data information. Refs. [31] introduced the design idea of the active data collection system and pointed out that the key to the collection system is task-driven, which is based on data types and experimental conditions. The design of the data collection system, for the task of DNS simulation on the cyber range, can be abstracted into 3 necessary steps: confirm the collected data, clean the invalid data, format and generate the data. The detailed explanation of these steps is as follows:

Step 1. **Confirmation of the collected data.** The data to be collected is divided into two types according to the importance: necessary data and optional data. *Necessary*: Actively obtain the data needed to construct the domain name resolution system. The lack of these data will make the system unable to build. The server of the domain name resolution system is composed of a series of domain name servers (recursive NS and authoritative NS). The main component of the resolution is the domain name data under the server. Therefore, the actively collected data includes two parts: NS data, domain name data. Meanwhile, in order to ensure the authenticity of the data, these data usually originate from authoritative public organizations. *Optional*: The IP

geographic location information of the host can be used to assist in the analysis of data transfer between different regions during the domain name resolution process.

Step 2. **Invalid data cleaning.** The data directly crawled can not be used as the final data, then often needed to filter. This is because any way of collecting data cannot guarantee that all collected data is correct. There will always be a small amount of invalid data among the whole collection, which can be explained by two reasons. First, the active measurement method collects data through request and response, and the network environment is constantly changing during this process. Second, the most realistic domain name data is time-sensitive, which means that the results of collecting the same domain name in different periods may be different.

Step 3. **Data formatting and generation.** Data formatting can be seen as a requirement for the server's storage. Data generation is usually a custom set of rules to create data, which is different from the data collection. *Formatting*: The DNS is a hierarchical structure. The realization of this structure is based on name server information and domain name data in the server. The available server and domain name data are organized into text-type files, which is usually called the zone file, dedicated to DNS construction according to the requirements of the DNS structure. *Generate*: The client requests domain name resolution, and the domain name server is responsible for responding to the results of domain name resolution. The CR provides a variety of virtual devices, and their IP addresses should be unique, so as to realize the simulation of the real network environment. To avoid virtual IP address conflicts between devices, it is also necessary to actively generate client IP data.

According to the above-mentioned reasons, we design a data generation system, which could flexibly and efficiently generate demanding DNS data, for quickly building a DNS simulation environment on the cyber range. Our system consists of three main components, as shown in **Fig. 1**: Collector, Processor, and Generator.

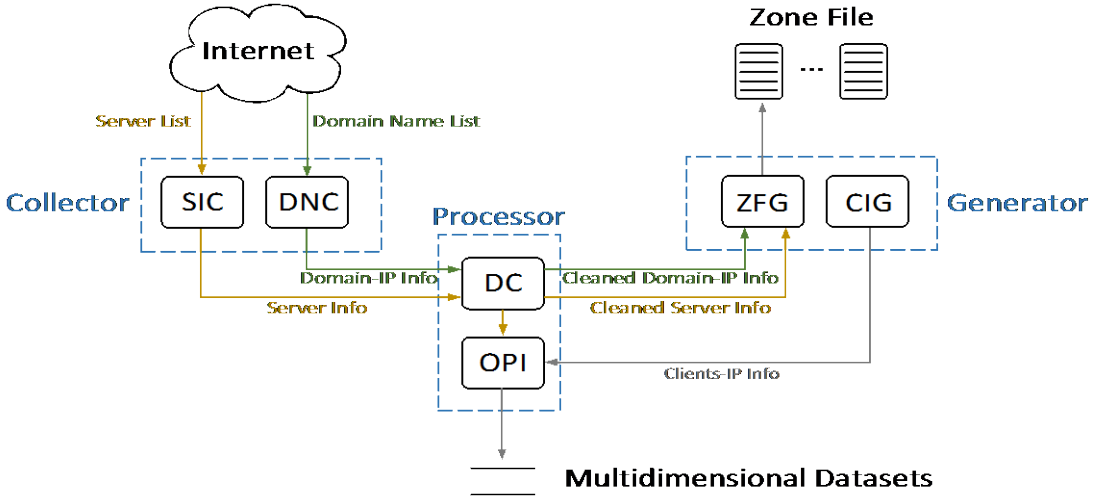


Fig. 1. System Architecture

- Collector: Obtaining the domain name lists and the IP addresses of the DNS servers which include name servers and recursive servers collected through DNS resolutions. This module includes two parts: Server Info Collector (SIC) and Domain Name Collector (DNC).

- Processor: Cleaning the collected data and personalizing it according to the demands. Processor consists of two parts, Data Cleaning (DC) and Optional Personalized Information (OPI). Our system initially incorporates a geographic information generation part.
- Generator: Generating the zone files and testing client information. Generator has two parts, Zone File Generator (ZFG) and Client Information Generator (CIG).

The scalability of our system architecture is very strong. These modules have been carefully split and the coupling between each module is very small. Each part of each module can be rewritten according to the requirements. In Section 4, we will introduce the design ideas and effects of these modules in detail.

Here is a brief description of the two output data generation processes: (1) Personalized Multidimensional Datasets Generation, (2) Zone File Generation.

Personalized Multidimensional Datasets Generation. First, the data obtained on the Internet is input into the collector, and the purpose of processing the data is to obtain some simple information, such as IP. Then, the output data of the collector is input into the processor, which will be filtered to get valid data and add personalized attributes. Finally, the output of the processor is multidimensional datasets.

Zone File Generation. As can be seen from **Fig. 1** the first two stages of the zone file generation process are generally consistent with the first two phases of the multidimensional datasets and are not described here. There are two kinds of data output by the DC part, namely Cleaned Domain-IP Info and Cleaned Server Info. The Server Info contains only the name server information. We input these two kinds of data into ZFG, and use the prepared area file template to generate area files in batches. This generation process is simple and fast.

4. Collector: Crawl Data

The Collector function is data crawling. It collects data mainly in two ways, that is, SIC collects domain name server data, and DNC collects domain name information.

4.1 Name Server Collector

This part is used to obtain DNS domain name server information, and the output is Server Info. DNS domain name servers can be divided into four types: root name server (root NS), top-level domain name server (TLD NS), recursive DNS server, authorization name server (authorization NS). To obtain these portable data, we gain these four types of data information in two ways: (1) download from the authoritative website to get the first three types of data. (2) query the authorization name server data to get the last one.

Download from the authoritative website. For the DNS system, the hierarchical structure is that the highest level of its hierarchical structure is the root, and the next one is the TLD. It is very important to collect information about the data in these two areas. The root zone files store a series of records of the root domain name, such as *NS records*, *A records*, etc., which also store these records of TLDs. To build the root zone files, we need to obtain this information in advance, that is root NS and TLD NS. Fortunately, InterNIC [33] operated by ICANN fully provides these zone data completely, which guarantees the authority and authenticity of the zone data. To further simplify the data, only part of the needed records is extracted. For a generated single record, the attributes are triples, including name, IP and name server type: root or TLD. For recursive DNS server data, we use a public DNS database [34], which contains public recursive DNS servers that are accessible by IPv4 or IPv6. Currently, there are 3,687 Nameservers from 239 countries in the database. For the purpose of data

balance and randomization, we indiscriminately selected the recursive DNS server data of multiple countries from this database, with each region selecting the same amount of data. Finally, for a single record, its attributes include *IP* and server type: *recursive*.

Obtain data by querying. In addition to preparing the data of the root zone files, we also need to build the zone files of the authorization NS, which include the information of domain name. To achieve this, our method is to query the SOA record of hot-spot second-level domain (SLD) Alexa [35], and then further process the obtained SOA name, that is, obtain the corresponding IP through DNS resolution. The data record generated at this time only contains *name*, *IP*, and name server type: *authoritative*. As for the current DNS, it is a kind of hierarchical structure, based on the three types of domain name server data that have been obtained: root NS, TLD NS, and authorization NS. Among them, the first two have built a hierarchical relationship from InterNIC open database, the latter two relationships are constructed by specifying the authorization NS to his parent TLD NS node, by adding a tag attribute "parent" to indicate the parent-child relationship.

4.2 Domain Name Collector

In this part, we first obtain the data of the domain name information from the open datasets of the Internet [36], which is grabbed from Alexa. Then we implemented a function that queries the IP address corresponding to the entered domain name list. The function calls the dig command to grab the DNS response packet and extract the IP address. Finally, it generates a file consisting of a series of domain names and IPs that will be a part of inputs for the next module. Here is an example of a record in the *domain-IP* file.

baidu.com 39.156.69.79

The course of correct DNS resolution depends on many conditions, e.g., the traffic environment is not blocked and the nameservers are working on its normal resolution path. Obviously, the resolution not always successful. In the process of query, we temporarily mark the domain name information that is not resolved correctly by judging the response code from the recursive server in the process of querying, if necessary, to analyze the proportion of incorrect and correct data. The correct information includes domain name and IP, while the Info that cannot be resolved normally marks with an error flag, which has four types: *Host Not Found (HNF), Type Not Found (TNF), Out of Time Error (OTE), and Server Fail (SF)*. The meaning of each error shows at **Table 1**. The incorrect domain name information can be filtered out in the next processor module.

Table 1. The Flag of Error DNS Resolution

Flag	Explanation
Host Not Found (HNF)	resolution query exceeded (limit: 5 time)
Type Not Found (TNF)	resolution type is not contained in the query result
Out of Time Error (OTE)	resolution query timeout (limit: 5 seconds)
Server Fail (SF)	uncategorized error types

5. Processor: Data Cleaning and Augmentation

The processor module, which contains both DC and OPI, is used for deep processing of data. The DC part is used to filter filtering out useless data, e.g., purify the outputs of the collector module, and then transfer the cleaned data sets to the generator module. The OPI section adds personalized properties to the nameserver datasets. Meanwhile, multidimensional data can also be used in a defined scene.

5.1 Data Cleaning

In the DC part, we focus on the actual requirements of the DNS simulation on the cyber range, and build a hierarchical but small-scale DNS resolution system. The purpose of our design is to make the final data more complete and more accurate, meanwhile, reduce the complexity of processing and make our system light-weighted. Therefore, our functions of the DC part further process the server Info and domain-IP Info that obtained in the collector module and delete the Info marked with any "error" flag. For example, the correct domain data resolution will be remained such as *google.com 216.58.200.238* and *ebay.com 66.135.195.175 66.135.196.249*, but the errors will be filtered out such as *theladbible.com TNF*. It is worth mentioning that retesting unresolved DNS data or fixing certain errors through a contrived filter rules may also be the choice of the DC part.

5.2 Optional Personalized Information

In the OPI part, the processed data can be added with additional attributes, which is discretionary. For example, we can use the IP address to query the geographical location information [37] (e.g. its latitude and longitude), which can be used in professional fields such as network mapping, e.g, *202.112.0.44* belongs to *Beijing CN* and the latitude and longitude is *39.906217 116.3912757* in the [37] database. It's trivial of OPI that can be completely independent of our system, but the reason for its retention is to focus on the integrity of DNS data processing and decouple our system from other modules in the simulation system.

6. Generator: Generate Final Data

The generator is used to generate data, which includes ZFG (Zone File Generator) and CIG (Client Info Generator). The ZFG generates zone files under each domain name server. And the CIG generates the client information.

6.1 Zone File Generator

In the ZFG part, the inputs are the information of processed domain name data and domain name server, which are used to build a simple hierarchy.

The content of each zone file is similar, so we use a series of template string to generate zone file. For example, "*NS_NAME TTL IN NS NS_SEVER*" with *NS_NAME*=".", *TTL=86400* and *NS_SEVER*="*a.root-servers.net.* " could be replaced to ". *86400 IN NS a.root-servers.net.*" as shown in **Fig. 2**.

```
.                                       86400       IN          NS          a.root-servers.net.
a.root-servers.net.         518400      IN          A           198.97.190.53

com.            86400       IN          NS          ns.com.
ns.com.         518400      IN          A           199.19.56.2

edu.            86400       IN          NS          ns.edu.
ns.edu.         518400      IN          A           37.209.198.9

gov.            86400       IN          NS          ns.gov.
ns.gov.         518400      IN          A           199.19.54.2

org.            86400       IN          NS          ns.org.
ns.org.         518400      IN          A           81.19.194.30
```

Fig. 2. A Zone File Example of Root

In reality, the physical deployment structure of the DNS domain name server is complex and redundant, as shown in **Fig. 3(a)**. The number of actual IP addresses of the "com" domain

name may be n (IP1 to IPn), and the number may be m as to the actual IP address corresponding to the "baidu.com" domain name (IP1 to IPm). There exist many-to-many relationships in the actual physical deployment of domain name services. For example, the IP1 of "baidu.com" has a corresponding relationship among IP1, IP2, and IPn of "com" and the IP2 of "baidu.com" has a corresponding relationship among IP1, IP2, and IPn of "com". These many-to-many relationships result in massive and redundant DNS domain name data, and it is difficult to filter useful information from the huge data. It is indeed an optimal ideal situation to completely imitate the real DNS structure, but it is not suitable for simulation on the cyber range. Because it has a relatively higher storage cost and the time cost of searching DNS records.

Following the above analysis, we design a simple and efficient hierarchical physical deployment structure that uses a "one-to-many" relationship to handle the IP correspondence of each subordinate domain name under its superior domain name as shown in **Fig. 3(b)**. For example, the name server of "com" contains many subdomains. Each subdomain name only has one IP, which is the same as "com". This design is meant to maintain the DNS hierarchical structure $R \rightarrow t \rightarrow a$ and obey the goals of light-weighted DNS data collection at the same time.

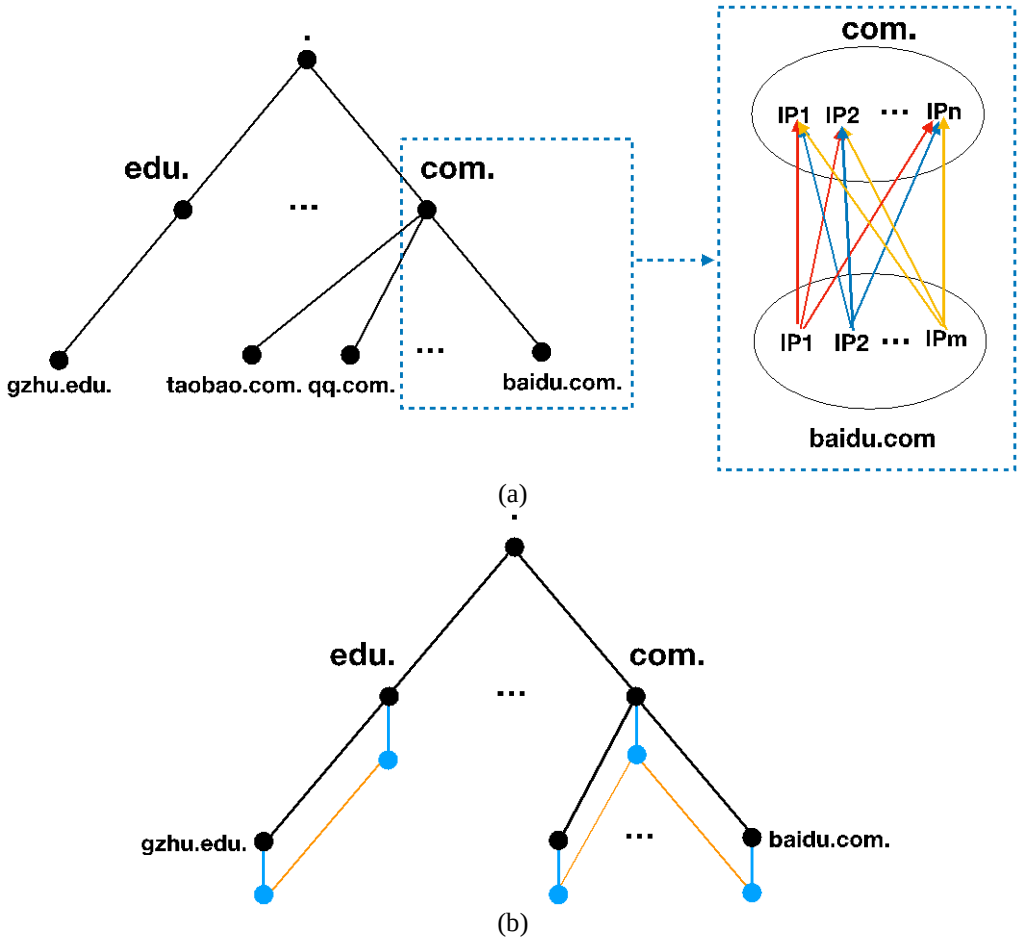


Fig. 3. (a): Complex DNS physical deployment structure; (b): Our DNS physical deployment structure

6.2 Client Info Generator

After the data has been processed by *Collector* and *Processor*, and the zone files have been created by *Zone File Generator*, a domain name resolution system has been naturally constructed. But in the actual situation, the construction of the resolution process of the DNS system starts when the client requests a recursive server, so the client information is essential. *Clients Info* must be generated to complete the entire DNS simulation system and make the DNS resolution works properly. In the CIG part, the client's IP address generation could be random or continuous. For example, the set of IPs *166.255.255.11~166.255.255.60* represents 50 clients. When generating the client's IP, we need to ensure that it does not conflict with the domain name and the name server, and assign a recursive NS to each client IP to make the request.

7. Application

In addition to collecting DNS datasets for network space mapping, this method can also accurately collect the required datasets according to the specific requirements of users, such as the simulation of DNS attacks and defense scenarios [32], the construction of new domain name resolution systems, etc.

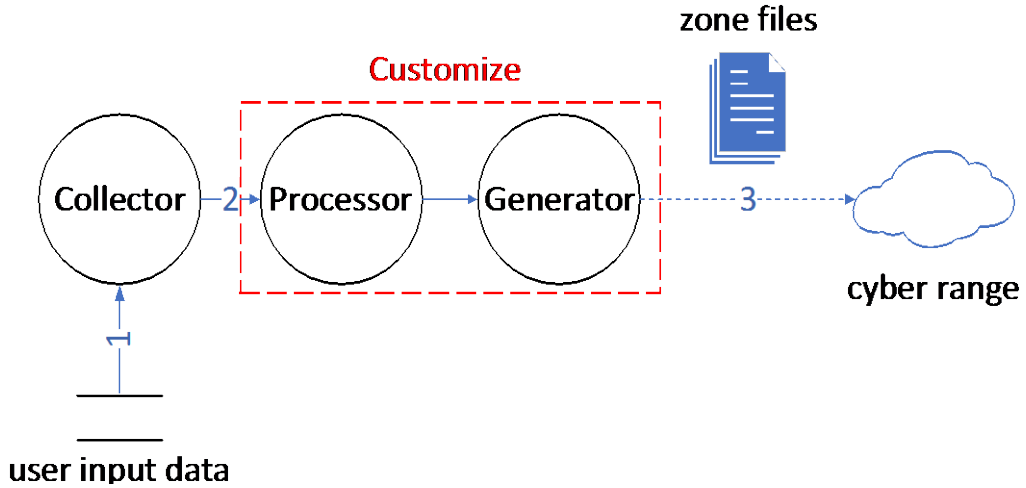


Fig. 4. The Simulation of DNS Attacks and Defense Scenarios

In order to implement these desired applications, the first step is to build a DNS resolution system on the CR, which can be virtually configured. Here, some routers are used to forward DNS software packages, and some switches are used for subnetting. For small scenarios, we use a star topology without routers. We use a single switch to divide enough VLANs, and put the servers and clients in the same broadcast domain so that the nodes can communicate with each other. For large scenarios, we use multiple switches to divide VLANs and configure some server nodes under each VLAN. At the same time, each switch is configured with a router, and each router is connected to all switches. DNS packets are forwarded through the router, which can reduce the load of each switch compared to the small scenario solution. For example, we distribute each container node with 1 core and 2G memory in small scenario. One of our configurations is as follows: $N_s(R,t,a,r,c)=(1,16,74,10,100)$ is the node number of servers and $N_Z(R,t,a)=(1,16,41484)$ is the number of zone files.

A. DNS attack and defense scenario simulation on the CR

The Domain Name System has been an essential component of the functionality of the Internet since 1985 [38], and the expansion of the Internet into the commercial sector enhances the requirements for security measures to protect data integrity and users' authentication. Before we strengthen the DNS security, the needed work is to simulate the scenario in which DNS works normally and perform various tests there. However, if we want to find the security issues that are close to our real network situation, it is not enough to simulate the scene by randomly generating the DNS data.

According to our method, it's easier to collect the datasets that can be more suitable for this scenario, as well as making simulation to proceed smoothly. As **Fig. 4** shows, this part is divided into the following three steps:

1. In the initial stage, the users first input the domain list into the processing module for preprocessing.
2. In the data processing and generation phase, the users write their own rules according to the specific requirements. Then the program generates the final DNS datasets according to the corresponding rules required for the simulation scenario, such as cyber range.
3. In the data usage stage, the users select the corresponding data combination according to the specific simulation requirements and deploys it into the specific attack scenario.

B. Construction of new domain name resolution systems on the CR

DNS translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. With the improvement of the network infrastructure, there may be new domain name resolution systems to solve new problems and replace the current DNS.

In order to smoothly upgrade the new domain name resolution systems from the current DNS, we need to store the current DNS datasets and put them into the new systems. Based on our method, we can accurately collect DNS data from the core nodes, so that the authorization and other works can be performed smoothly.

8. Evaluation

In this part, we evaluate the lightness and efficiency of our data collection method. In Section 2, we point out that the purpose of some projects is to collect a large amount of data, which requires a lot of resource consumption. For example, the LDplayer collected 2.67TB in total and the Thales system consists of two parts, including 64 processing cores and 164GB of RAM in each of its two main parts. And these are difficult to apply under CR with limited resources. Therefore, we reflect the lightness of our method by evaluating the data storage space consumption and program running time as follows. The first one is the relationship between the scale of deployment with the number of files and storage capacity in the name server zone. The second is the relationship between the amount of domain name and time of data generation, and the analysis of the main factors affecting the time of data generation.

8.1 Experimental Environment and Setup

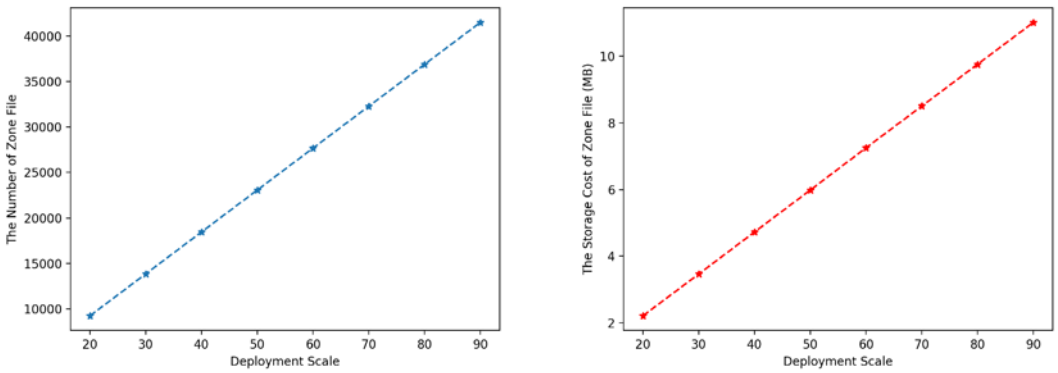
Our experimental environment includes: a host running a data collection program with 8GB of memory, 4 cores and 8 CPU threads, SSD disk, and CentOS 7 as the operating system. For the network environment, the uplink speed is 20~30Mbps and the downlink speed is 30~40Mbps. The parameter settings of the data collection program mainly include: the number of running

threads is 8, the upper bound of the resolution time is 5s, and the number of resolutions retries is 3.

8.2 Zone Files and Storage that meet the Scale Requirements

We formally defined the requirements of the DNS simulation system on the CR in Section 3.1, a 5-tuple (R,t,a,r,c), and the size of the servers deployment scale $N_S(R,t,a)$ that determines the amount of zone file $N_Z(R,t,a)$ required. **Fig. 5** reflects a visual description of this relationship. As the number of deployed servers $N_S(R,t,a)$ increases, the total number of zone files $N_Z(R,t,a)$ also increases. It can be seen that they have a linear positive correlation, which is mainly caused by two reasons: 1) The total number of zone files depends on the number of authoritative servers a, and authoritative servers a account for the majority of simulation system. 2) Our actual system deployment is to evenly distribute zone files, which can make DNS access as dispersed as possible.

Furthermore, as shown in **Fig. 5(a)**, the storage capacity change reflects the deployment scale $N_S(R,t,a)$. We can see that at the highest point $N_S(R,t,a)=41484$, although the name server has $N_S(R,t,a)=90$ devices. The total storage capacity just occupies $S(R,t,a)=12.7(MB)$ as shown in **Fig. 5(b)**. This is because the information contained in each zone file is the most concise (only *A* record). We can statistically observe that the size of the zone file fluctuates in the range of 200 bytes to 300 bytes, which also reflects the lightness of our data generation method.



(a) (b)

Fig. 5. (a): Changes in The Number of Zone File; (b): Changes in The Storage Cost of Zone File

8.3 Time Performance Analysis

In order to analyze the time performance of our method, we have tested 25 sets of domain name data resolution requests with a range of 20 to 140,000, as shown in **Fig. 6.** As the number of domain names increases, the time-consuming process of the program shows an increasing trend. It is about 7 hours when the number of domain is 140,000.

We divide the image into three parts and analyze the main causes of this trend: 1) For domain number smaller than 2,500, the time consumption increases rapidly. This is because for a small number of domain names, the time consumption is greatly affected by network

fluctuations; 2) For domain number between 2,500 and 30,000, time consumption increases at a slower rate. Because the resolution response speed will be faster when the domain name is cached by the recursive server. It should be noted that the recursive server can only cache a certain number of domain names; 3) For domain number greater than 30,000, the time cost of our program increases linearly without obvious fluctuations. The reason is that the amount of data is large, and influencing factors such as network delay are also alleviated.

It is worth mentioning that the *Collector* module accounts for costing the most time in our system. As given in Section 3.2, our system is divided into three modules: *Collector*, *Processor*, and *Generator*. Among them, the order of importance that affects the entire system time is: $Collector \gg Generator \geq Processor$. In order to collect *domain-IP* Info, the former module needs to perform DNS resolution request operations, which includes network communication and disk I/O. The last two modules mainly involve disk I/O operations to generate zone files. Generally, network communication operations take more time than disk I/O. In Section 4.2, we have classified the abnormal resolution that may occur in the collection of domain-IP Info, which including four types. These abnormalities are the main factors affected our program performance. Conversely, reducing these abnormalities can promote and stabilize program performance. Following the above discussion, we have conducted a deeper investigation, as shown in **Fig. 7**. This means that in addition to the previous incremental analysis of domain names, the time performance of each 10,000 domain names is also analyzed separately.

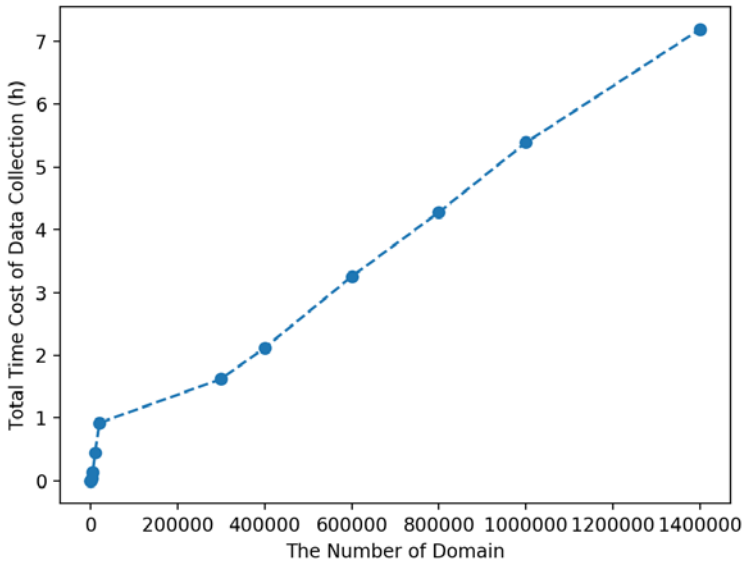


Fig. 6. The Number of Domain vs Total Time Cost

These are non-duplicate domain names randomly selected from 140,000 domain names. In a series of tests, we test 14 sets of data, and each set of tests had 10,000 domain names. The group number of these tests can be defined as g_i $(i = 0,1,\ldots,13)$, and $T(g_i)$ (unit: hour) represents the time required for group g_i to run. The smaller the value of $T(g_i)$, the faster

the program runs. $N(g_i)$ means the number of abnormal data in the g_i group. The smaller the value of $N(g_i)$, the less abnormal data.

Fig. 7(a) reflects that the running time of the program fluctuates greatly. Among them, the value of $T(g_{10})$ and $T(g_{12})$ is smaller, indicating that their running time is faster. In particular, g_{12} is the fastest, and $T(g_{12})=0.367$. Similarly, $T(g_4)$, $T(g_5)$, $T(g_8)$ and $T(g_9)$ have larger values, where g_5 is the slowest and $T(g_5)=0.569$. **Fig. 7(b)** shows the anomaly resolution data contained in each group of tests. $N(g_{10})$ and $N(g_{12})$ are smaller, and g_{10} contains the least abnormal data, where $N(g_{10})=2020$. $N(g_5)$ and $N(g_9)$ have larger values, indicating that they have more abnormal data. Wherein, g_9 has the most abnormal data, and $N(g_9)=2709$.

$$R_t(g_i)=\frac{100\cdot T(g_i)}{\sum_{i=0}^{13}T(g_i)}, R_a(g_i)=\frac{100\cdot N(g_i)}{\sum_{i=0}^{13}N(g_i)} \tag{1}$$

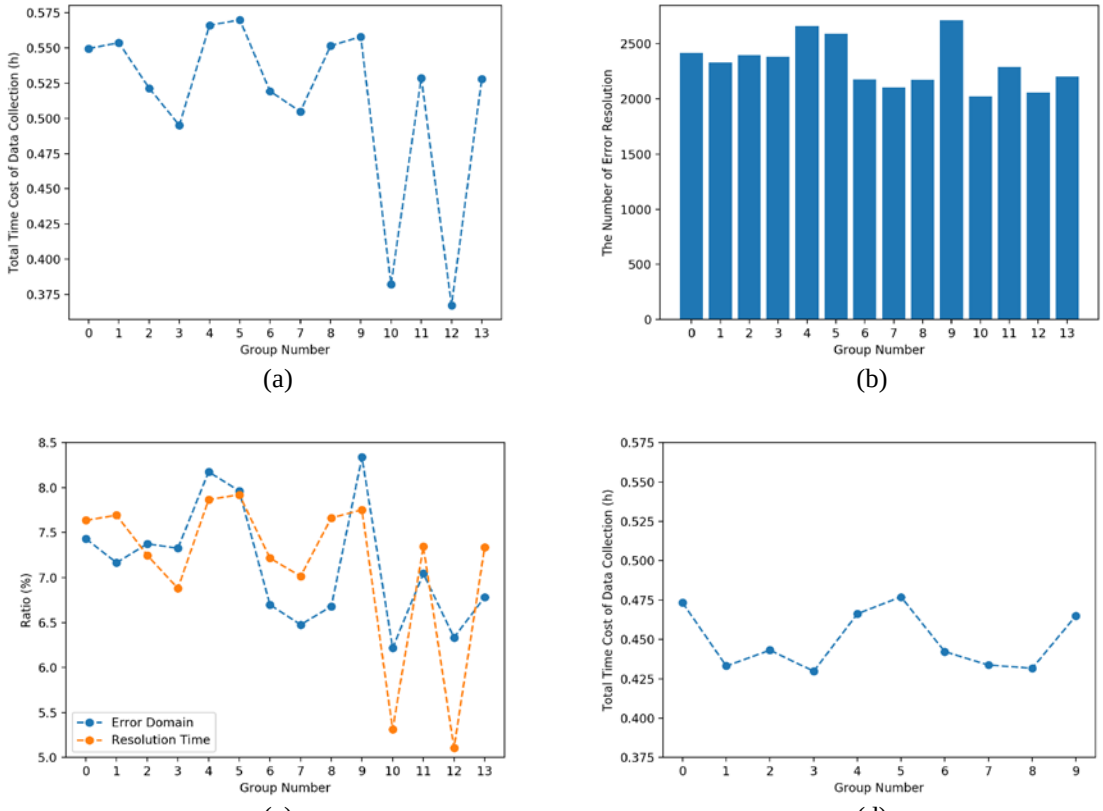


Fig. 7. (a): The time cost of each 10,000 domain resolutions; (b): The error number of each 10,000 domain resolutions; (c): The error and time cost ratio of resolutions; (d): The time cost of normal resolutions

We define $R_t(g_i)$ as the percentage of time $T(g_i)$ cost by each group g_i in the total time spent, and define $R_a(g_i)$ as the percentage of the abnormal data $N(g_i)$ contained in each group g_i in the total abnormal data. The two ratio are calculated by the formula (1). **Fig. 7(c)** shows the calculation results of $R_t(g_i)$ and $R_a(g_i)$. From the perspective of the tendency of the curve, the changes of $R_t(g_i)$ and $R_a(g_i)$ tend to be consistent. The more abnormal data, the longer the program runs. In other words, the running time of the data collection program is closely related to the amount of abnormal data. Therefore, we believe that reducing abnormal data can improve program performance.

Finally, we test 10 groups of data, each set of data also has 10,000 domain names, but we only contain correctly resolved data. We can intuitively draw from **Fig. 7(d)** that the program running time is relatively stable, which is less than the running time of the test examples in **Fig. 7(a)**. After calculation, the program performance has improved by 12.5% on average by removing the abnormal data.

9. Conclusion

Data collection is the first step in the DNS simulation, and the quality of the datasets can have a profound impact on the experimental results. To handle the various issues in the DNS simulation, we propose an active and light-weighted collection method about DNS data. Then, we discuss the application of our data collection method in two aspects and explain the practicality of it. Finally, we list plenty of charts to show the time and storage cost of the data collection method, and infer the efficiency of the method through in-depth exploration. In the future, we will consider adjusting the design of modules to make our method suitable for DNS simulations for other purposes.

Acknowledgment

This work was supported in part by National Key research and Development Plan (Grant No. 2018YFB0803504),the Guangdong Province Key Research and Development Plan (Grant No. 2019B010137004), the National Natural Science Foundation of China (Grant No. U1636215, 61572492, 61902083, 61976064), and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019) and Zhijiang International Young Talent Scheme (2019).

References

[1] P. V. Mockapetris, “Domain names: Implementation specification,” 1983.
[2] J. Postel, “Internet protocol—DARPA internet program protocol specification, rfc 791,” 1981.
[3] Y. Zhang, “An autonomous open root resolution architecture for domain name system in the internet,” *Journal of Cyber Security*, vol. 2, no. 4, pp. 57–69, 2017. Article (CrossRef Link)
[4] Q. Tan, Y. Gao, J. Shi, X. Wang, B. Fang, and Z. Tian, “Toward a comprehensive insight into the eclipse attacks of tor hidden services,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1584–1593, 2018. Article (CrossRef Link)
[5] Z. Tian et al., “Real time lateral movement detection based on evidence reasoning network for edge computing environment,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4285-4294, 2019. Article (CrossRef Link)

[6] Z. Tian, M. Li, M. Qiu, Y. Sun, and S. Su, “Block-def: A secure digital evidence framework using blockchain,” *Information Sciences*, vol. 491, pp. 151–165, 2019. Article (CrossRef Link)
[7] L. Zhu and J. Heidemann, “LDplayer: DNS experimentation at scale,” in *Proc. of the internet measurement conference 2018*, pp. 119–132, 2018. Article (CrossRef Link)
[8] A. Kountouras et al., “Enabling network security through active dns datasets,” in *Proc. of International symposium on research in attacks, intrusions, and defenses*, pp. 188–208, 2016. Article (CrossRef Link)
[9] H.-L. ZHANG, B.-X. FANG, M.-Z. HU, Y. JIANG, C.-Y. ZHAN, and S.-F. ZHANG, “A survey on internet measurement and analysis,” *Journal of Software*, vol. 14, no. 1, pp. 110–116, 2003.
[10] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, “Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6266-6278, 2020. Article (CrossRef Link)
[11] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, “A distributed deep learning system for web attack detection on edge devices,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963–1971, 2020. Article (CrossRef Link)
[12] Weimer, Florian, "Passive DNS replication," in *Proc. of FIRST conference on computer security incident*, p. 98. 2005.
[13] Zdrnja, Bojan, Nevil Brownlee, and Duane Wessels, "Passive monitoring of DNS anomalies," in *Proc. of International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 129-139, 2007. Article (CrossRef Link)
[14] Plonka, David, and Paul Barford, "Context-aware clustering of DNS query traffic," in *Proc. of the 8th ACM SIGCOMM conference on Internet measurement*, pp. 217-230, 2008. Article (CrossRef Link)
[15] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee, “DNS noise: Measuring the pervasiveness of disposable domains in modern dns traffic,” in *Proc. of 2014 44th annual ieee/ifip international conference on dependable systems and networks*, pp. 598–609, 2014. Article (CrossRef Link)
[16] L. Yin, Y. GUO, H. ZHANG, W. HUANG, and B. FANG, “Threat-based declassification and endorsement for mobile computing,” *Chinese Journal of Electronics*, vol. 28, pp. 1041–1052, Sep. 2019. Article (CrossRef Link)
[17] Durumeric, Zakir, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman, "A search engine backed by Internet-wide scanning," in *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542-553, 2015. Article (CrossRef Link)
[18] D. Herrmann, C. Banse, and H. Federrath, “Behavior-based tracking: Exploiting characteristic patterns in dns traffic,” *Computers & Security*, vol. 39, pp. 17–33, 2013. Article (CrossRef Link)
[19] J. Oberheide, M. Karir, and Z. M. Mao, “Characterizing dark dns behavior,” in *Proc. of International conference on detection of intrusions and malware, and vulnerability assessment*, pp. 140–156, 2007. Article (CrossRef Link)
[20] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z. Zhang, “Identifying suspicious activities through dns failure graph analysis,” in *Proc. of The 18th ieee international conference on network protocols*, pp. 144–153, 2010. Article (CrossRef Link)
[21] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, and A. Nucci, “DNS to the rescue: Discerning content and services in a tangled web,” in *Proc. of the 2012 internet measurement conference*, pp. 413–426, 2012. Article (CrossRef Link)
[22] A. M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi, “Detection of malicious payload distribution channels in dns,” in *Proc. of 2014 ieee international conference on communications (icc)*, 2014.
[23] K. Schomp, M. Allman, and M. Rabinovich, “DNS resolvers considered harmful,” in *Proc. of the 13th acm workshop on hot topics in networks*, pp. 16:1–16:7, 2014. Article (CrossRef Link)
[24] Z. Tian, S. Su, W. Shi, X. Du, M. Guizani, and X. Yu, “A data-driven method for future internet route decision modeling,” *Future Generation Computer Systems*, vol. 95, pp. 212–220, 2019. Article (CrossRef Link)

[25] P. Foremski, C. Callegari, and M. Pagano, “DNS-class: Immediate classification of ip flows using dns,” *International Journal of Network Management*, vol. 24, no. 4, pp. 272–288, 2014. Article (CrossRef Link)
[26] Z. Tian, X. Gao, S. Su, J. Qiu, X. Du, and M. Guizani, “Evaluating reputation management schemes of internet of vehicles based on evolutionary game theory,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5971-5980, 2019. Article (CrossRef Link)
[27] Z. Tian, X. Gao, S. Su, and J. Qiu, “Vcash: A novel reputation framework for identifying denial of traffic service in internet of connected vehicles,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3901-3909, 2020. Article (CrossRef Link)
[28] J. Qiu, L. Du, D. Zhang, S. Su, and Z. Tian, “Nei-tte: Intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2659–2666, 2020. Article (CrossRef Link)
[29] L. Yin, X. Luo, C. Zhu, L. Wang, Z. Xu, and H. Lu, “ConnSpoiler: Disrupting c&c communication of iot-based botnet through fast detection of anomalous domain queries,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1373–1384, Feb. 2020. Article (CrossRef Link)
[30] J. Davis and S. Magrath, “A survey of cyber ranges and testbeds,” *DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA) CYBER AND*, 2013.
[31] Laflamme, Francois, Mike Maydan, and Andrew Miller. "Using paradata to actively manage data collection survey process," in *Proc. of the Survey Research Methods Section, American Statistical Association*, 2008.
[32] S. K. Damodaran and J. M. Couretas, “Cyber modeling & simulation for cyber-range events,” in *Proc. of the conference on summer computer simulation*, pp. 1–8, 2015.
[33] “InterNIC.” https://www.internic.net/, 2019.
[34] “Public DNS Server List.” https://public-dns.info/, 2019.
[35] “Alexa.” https://www.alexa.com/, 2019.
[36] “Alexa domain list.” https://github.com/fgont/domain-list, 2017.
[37] “IP to City Lite database.” https://db-ip.com/db/download/ip-to-city-lite, 2019.
[38] “Domain Name System.” https://en.wikipedia.org/wiki/Domain_Name_System.

Shuang Li, born in 1997, received her B.E. degree in information security from Harbin Institute of Technology in 2019 and is currently pursuing the master degree in cyberspace security from Guangzhou University. Her research interests include blockchain, deep learning, computer network and DNS.

Shasha Du, received the B.E. degree in software engineering at Central South University, Changsha, China, in 2018. She is currently pursuing the M.E. degree in computer technology at Guangzhou University, Guangzhou, China. Her research interest includes blockchain, computer network and DNS.

Wenfeng Huang, received the B.E. degree in telecommunication engineering at Northeastern University, Shenyang, China, in 2019. He is currently pursuing the M.E. degree in computer technology at Guangzhou University, Guangzhou, China. His research interests: deep learning, border gateway protocol, computer network and DNS.

Siyu Liang (1995 -), male (Han) master, GuangZhou University, research interests: blockchain, DNS, network security.

Jinxi Deng received his BS degree in Computer Science and Technology from South China University in 2018. And he is currently completing a MS in Computer Science and Technology in Guangzhou University. His research interest includes DNS, Blockchain and software engineering.

LE WANG received the Ph.D. degree in computer science from the National University of Defense Technology (NUDT). He is currently an Associate Professor with the Cyberspace Institute of Advanced Technology, Guangzhou University, and a Postdoctoral Student with the Computer School, NUDT. His current research interests include network and big data security. He is a member of the China Computer Federation.

Huiwu Huang (1974 -), male (Han) master, Guangdong University of technology, research interests: computer network, network security.

Xinhai Liao (1976 -), male (Han) master, senior engineer from Chenzhou, Hunan Province. His research interests: computer network security and application, education informatization.

Shen Su, he got his PH. D from Harbin Institute of Technology. His research interest includes blockchain security, DNS, vehicular network, and route modeling.