International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

DNS ANY Request Cannon Activity in DNS Query Packet Traffic

Yuto Takeda[1], Yasuo Musashi[2*], Kenichi Sugitani[2], and Toshiyuki Moriyama[3]

[1]*Graduate School of Science and Technology, Kumamoto University,*
2-39-1 Kurokami, Central W. Kumamoto, JAPAN, 860-8555
[2]*Center for Multimedia and Information Technologies, Kumamoto University,*
2-39-1 Kurokami, Central W. Kumamoto, JAPAN, 860-8555
[3]*Faculty of Social and Environmental Studies, Fukuoka Institute of Technology,*
3-10-1 Wajirohigashi, East W., Fukuoka, JAPAN, 842-0295
* Corresponding author's Email: musashi@cc.kumamoto-u.ac.jp

Abstract: We statistically investigated the total ANY resource record (RR) based DNS query request packet traffic from the Internet to the top domain DNS server in a university campus network from January 1st, 2011 to December 31st, 2012. The obtained results are: (1) we found a significant increase in the inbound ANY RR based DNS query request traffic at November 28th, 2011. (2) In the DNS query request packet traffic, we observed only a query keyword of the campus domain name. (3) We found a correlation between the total inbound DNS query request packet traffic and the DNS query request packet traffic including the query keyword. (4) We also carried out the loading test sending ANY, A, and PTR RRs based unique DNS queries to a test DNS server, however, we observed no difference among the vmstat parameters based on the queries, and the load value was only 0.10-0.20. These results indicate that the ANY RR based DNS request packet traffic is quite strange. Therefore, we should pay much attention to the ANY RR based DNS query request traffic including the single domain name.

Keywords: DNS ANY request cannon; DNS log analysis; DoS attack; DNS query loading test.

1. Introduction

It is of considerable importance to protect the domain name system (DNS) services and/or servers in the Internet, since they are very important components of the Internet infrastructures. If the DNS services stop, most the network applications or services will crash.

Recently, we observed interesting traffic bumps of the ANY resource record (RR) based DNS query request packet access to the top DNS (DNS) server in a university campus network continuously since November 28th, 2011. The traffic bumps have been also reported in the several Weblog sites [1, 2]. This is probably because the ANY RR based DNS query request packet access can perform or induce the DNS amplification attack employing the source IP address spoofing technology [3-5]. Therefore, it is very important to detect/prevent the ANY RR based DNS query request packet access to the DNS servers.

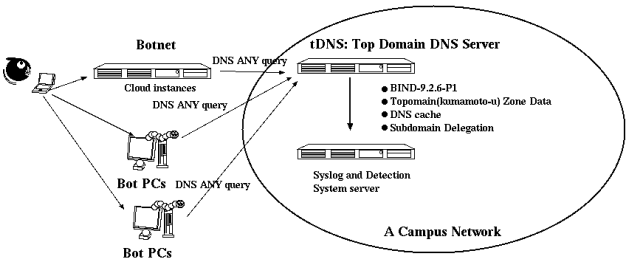

Figure 1 A schematic diagram of a network observed in the present study.

Previously, we reported development and evaluation of the restricted Damerau-Levenshtein [6, 7] distance based on detection model of the Kaminsky DNS cache poisoning attack in the total inbound A RR based DNS query request packet traffic to the

campus tDNS server from January 1st to December 31st, 2010 [8], and it can be also useful for detecting the ANY RR based DNS query request packet access.

In this paper, (1) we carried out restricted Damerau-Levenshtein distance based on the analysis on the total ANY resource record (RR) based DNS query request packet traffic from the Internet from January 1st, 2011 to December 31st, 2012. (2) In order to compare the load differences among the ANY, A, and PTR RRs based DNS query request packet traffic, we assessed the results for the query keywords in the ANY-RR based DNS query request packet traffic, and (3) performed the loading test sending ANY, A, and PTR RRs based DNS query request packet traffic which include the unique DNS query keywords to a test DNS server and employ the Atmel ATmega328P-PU microcontrollers [9] and the WIZnet W5100/W5200 chip based Ether interface modules [10].

2. Observation

2.1 Network systems and DNS query packet capturing

We investigated the DNS query request packet traffic between the top domain DNS (DNS) server and the DNS clients. Figure 1 shows an observed network system in the present study, which consists of the DNS server and the PC clients and/or the cloud instances as bots carrying out the DNS ANY Request Cannon (DARC) activity in the Internet. The DNS server is one of the top level domain name (kumamoto-u) system servers and it plays an important role of domain name resolution services including DNS cache functions, and subdomain name delegation services for many PC clients and the subdomain network servers. Its operating system is Linux OS (CentOS 6.4 Final) in which the kernel-2.6.32 is currently employed with the Intel Xeon X5660 2.8 GHz 6 Cores dual node system, the 16GB core memory, and Intel Corporation EthernetPro 82575EB Gigabit Ethernet Controllers.

In the DNS server, the BIND-9.8.2 program package has been employed as a DNS server daemon [11]. The DNS query request packets and their query keywords from the DNS clients have been captured and decoded by a query logging option (see Figure 2 and the named.conf manual of the BIND program in more detail). The log of DNS query request packet access has been recorded in the syslog files. All of the syslog files are daily updated by the cron system.

```
logging {
   channel qlog {
         syslog local1;
   };
   category queries { qlog };
}
```

Figure 2 DNS query request logging option in "named.conf" for BIND program packages.

```
Oct 12 08:38:24 kun named[533]: client 133.95.xxx.yyy#39815: query: 130.13.194.xxx.in-addr.arpa IN PTR
Oct 12 08:38:25 kun named[533]: client 133.95.xxx.yyy#39815: query: dmea.net IN MX
Oct 12 08:38:43 kun named[533]: client 133.95.xxx.yyy#39815: query: mxwall03.hkabc.net IN A
Oct 12 08:38:50 kun named[533]: client 133.95.xxx.yyy#39815: query: kumamoto-u.ac.jp IN ANY
```

Figure 3 Structure of syslog messages generated by BIND program packages.

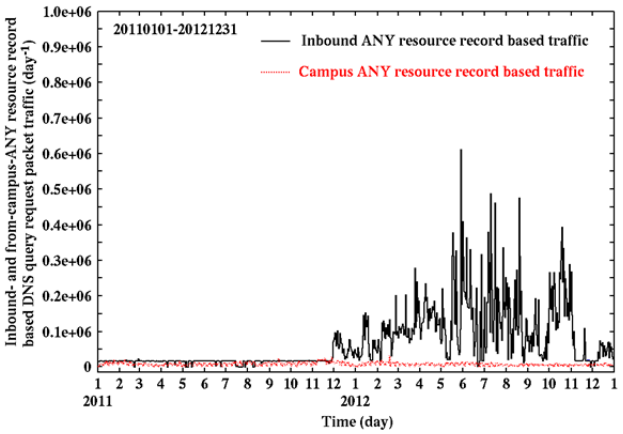

Figure 4 Changes in the ANY resource record based DNS request packet traffic from the campus network and the Internet. (day^{-1} unit).

The line of syslog message consists of the contents in the DNS query request packet like a time, a source IP address of the DNS client, a query keyword, a type of resource record (PTR, MX, A or ANY). There are several types of the resource record (RR) like a fully qualified domain name (A RR) type, an IP address (PTR RR) type, an E-mail exchange (MX RR) type, or an ANY RR type (see Figure 3).

2.2 Observed ANY resource record based DNS query request packet traffic

Firstly, we demonstrate the observed ANY resource record (RR) based DNS query request packet traffic from the campus network and the Internet to the top DNS (DNS) server from January 1st, 2011 to December 31st, 2012, as shown in Figure 4.

In Figure 4, we can observe that both traffic curve changes in a mild manner (the inbound traffic:

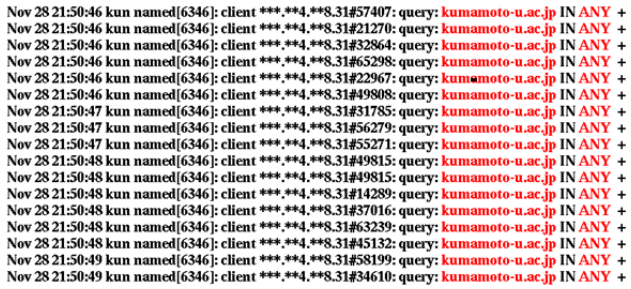

Figure 5 Changes in the DNS query keywords in the total ANY resource record (RR) based DNS query request packet traffic from the campus network to top domain DNS (DNS) server at November 28th, 2011.

18,000 day^{-1}, the traffic from the campus: 7,000 day^{-1}). However, we can see that the inbound ANY RR based DNS query request packet traffic drastically changes since November 28th, 2011. Daly reported the same bumps in the ANY RR based DNS query request packet traffic [1] and Shortt also called the traffic bumps a DNS ANY Request Cannon [2].

We also investigated the query keyword changes in the ANY RR based DNS query request packet traffic through the day of November 28th, 2011, and the results are shown in Figure 5.

In Figure 5, we can observe a continuously repeated sequence of the same query keyword "kumamoto-u.ac.jp". This feature shows that there an be a possibility to detect the inbound ANY RR based DNS query request packet traffic bumps more efficiently.

2.3 Detection model for DNS ANY Request Cannon

Here we define a detection model of the DNS ANY Request Cannon [2] or a traffic bump in the ANY RR based DNS query request packet access.

— A detection model — the DNS ANY Request Cannon (DARC) activity can be mainly carried out by a small number of IP hosts on the Internet or like the bot compromised PCs or the public cloud instances. Since these IP hosts send a lot of the ANY RR based DNS query request packets to the top domain DNS server, the traffic can be detected by calculating the Euclidian distance between the source IP addresses. So we suggest the restricted Damerau-Levenshtein (edit) distance [6, 7] based-detection system of the DARC activity, since the DARC activity causes the continuously repeated sequence of the same query keyword (Figure 5).

Here, we should also define thresholds for detecting the DARC activity, as setting to 10 packets day^{-1} for the frequencies of the top unique source IP addresses and for the edit distance, respectively.

2.4 Euclidian distance of source IP addresses

The Euclidean distances, ed(IP_i, IP_{i-1}), are calculated, as

$$ed(IP_i, IP_{i-1}) = \sqrt{\sum_{j=1}^{4} (x_{i,j} - x_{i-1,j})^2} \quad (1)$$

where both IP_i and IP_{i-1} are the current IP address i and the last IP address i-1 of the DNS query keywords, respectively, and where $x_{i,1}$, $x_{i,2}$, $x_{i,3}$, and $x_{i,4}$ correspond to an IPv4 address like A.B.C.D, respectively. For instance, if an IP address is 192.168.1.1, the vector $(x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4})^T$ can be represented as $(192.0, 168.0, 1.0, 1.0)^T$.

If the DARC activity model follows a single or distributed source IP address based-model i.e. we define the DARC activity, the detection is decided by thresholds $sd_{min}=sd_{max}=0.0$ or $sd_{min}=1.0$, $sd_{max}=5.0$ [12], as

$$sd_{min}(=0.0) \le ed(sIP_i, sIP_{i-1}) \le sd_{max}(=0.0) \text{ or}$$
$$sd_{min}(=1.0) \le ed(sIP_i, sIP_{i-1}) \le sd_{max}(=5.0) \quad (2)$$

2.4 Estimation of Damerau-Levehnshtein distance between domain names as query keywords

The Levenshtein (Edit) distance, LD (X, Y), is calculated, as

$$LD[x, y] = \min(LD[x-1][y]+1, LD[x][y-1]+1, LD[x-1][y-1]+cost) \quad (3)$$

both x and y are lengths of the strings X and Y, and the X and the Y are strings of the current domain name (DN) i and the last DN i-1 of the DNS query keywords, respectively. For instance, if the DNs are X = "a001.example.com" and Y = "a002.example.com", the Levenshtein distance LD (X,Y) is calculated to be 1, since the Levenshtein distance counts the number of edit operations like "insertion," "deletion," and "substitution" [6]. Furthermore, the restricted Damerau-Levenshtein distance takes into consideration the operation "transposition" in order to suppress the over-estimation [7]. The detection of the DARC activity is decided by thresholds $dl_{min}=dl_{max}=0$,

$$dl_{min} \le LD(DN_i, DN_{i-1}) \le dl_{max} \quad (4)$$

This is because the DARC activity causes the continuously repeated sequence of the same query

```
1  #!/bin/tcsh -f
2  set Threshold=10
3  # Step 1 Learning to produce a low-dimensional
4  cat /var/log/querylog | clgrep -v -cclients.conf | \
5  grep "IN ANY +" | \
6  sdis 0.0 0.0 1.0 5.0 | dlevens 0 0 | tr '#' ' ' | \
7  awk '{print $7}' | sort -r | uniq -c | sort -r | \
8  awk '{printf("%s\t%s\n",$2,$1);}' | \
9  qdos Threshold >tmpfile
10 # Step 2 Detection
11 cat /var/log/querlog | clgrep -ctmpfile | \
12 grep "IN ANY +" >ANYActDet.log
13 # Step 3 Scoring
14 cat ANYActDet.log | wc -l >>ANYActDetScore.txt
15 exit 0
```

Figure 6 Detection algorithm of DNS ANY Request Cannon (DARC) activity.

keyword (See Figure 5).

2.5 Detection algorithm for DARC activity

We suggest the following detection algorithm of the DNS ANY Request Cannon (DARC) activity and we show a prototype program (see Figure 6):

── **Step 1** *Learning to produce a low-dimensional* ─In this step, the **clgrep**, **cngrep**, and **grep** commands extract inbound ANY RR based DNS query request packet messages from the DNS query log file *(/var/log/querylog*) with discarding case-insensitively keywords local and *kumamoto-u*, the **sdis** command prints out a syslog message if the Euclidean distance of two source IP addresses is calculated to be zero or to take a range of 1.0-5.0 [12], the **dleven** command prints out the syslog message if the restricted Damerau-Levenshtein distance LD(DN_i, DN_{i-1}) takes a zero value (as discussed in the Section 2.4), and the **awk**, **sort**, **uniq**, and **qdos** commands (lines 7 to 9 in Figure 4) compute and check the frequencies of the restricted Damerau-Levenshtein distance LD(DN_i, DN_{i-1}) and if the frequency exceeds a threshold value (*Threshold*=10), they write out the candidate IP addresses into a *tmpfile* as training data.

── **Step 2** *Detection* ─ In the next step, the **clgrep** and **grep** commands extract messages related to the DNS ANY Request Cannon (DARC) activity in the DNS query request packet log file *(/var/log/querylog*), by using the training data (*tmpfile*) generated in the previous step and they generate only a DNS query log file (*ANYActDet.log*) related to DARC activity.

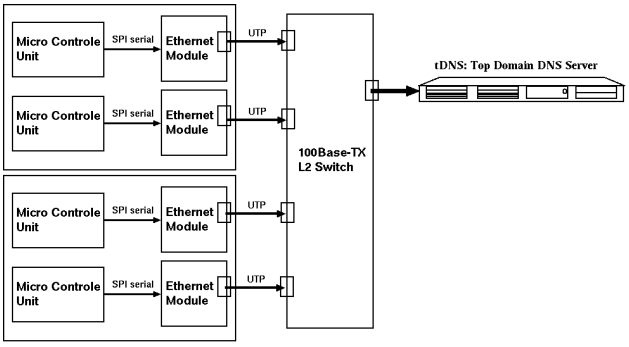

Figure 7 The structure block diagram for the DNS query packet traffic generating system (DQRPTGS).

── **Step 3** *Scoring* ─ In the final step, the **cat** and **wc** command calculates the score for the detection of the DARC activity in the file ANYActDet.log, and it writes out the detection score into a score file (ANYActDetScore.txt) in an appending manner.

2.6 DNS Query request packet traffic generating system

We developed the DNS query request packet traffic generating system (DQRPTGS) to perform loading tests generating ANY, A, and PTR RRs based DNS query request packet traffic including unique DNS query keywords to a test DNS server in the campus network, as shown in Figure 7.

The system consists of three parts: four micro-controller units (MCUs), four Ethernet modules (EMs), and an L2 switching HUB (L2-SW) with an uplink port and four 10/100Base-TX auto-negotiation ports.

In the MCUs, we employed the Atmel 8-bit ATmega328P-PU 28 pin DIP microcontroller unit with 32KB In-System Programmable Flash program memory [9], which works in 8 MHz, has a 2KB SRAM, an SPI serial interfaces, and uses an *optiboot* boot loader [13].

We deployed the WIZnet WIZ820io Ethernet module [14] for Ethernet interface, which consists of the WIZnet W5200 Fast SPI Ethernet Controller (Ethernet MAC/PHY embedded, 10Base-T/100Base-TX) chip, a transformer, and an RJ45 modular connector, and it can be easily controlled by the MCU with Serial Peripheral Interface (SPI Mode 0, 3).

We used the Arduino Integrated Development Environment (IDE: arduino-1.0.3) [15] to develop a DNS query request packet generator program for the MCU and the Ethernet module. The DNS query packet generating algorithm is shown as:

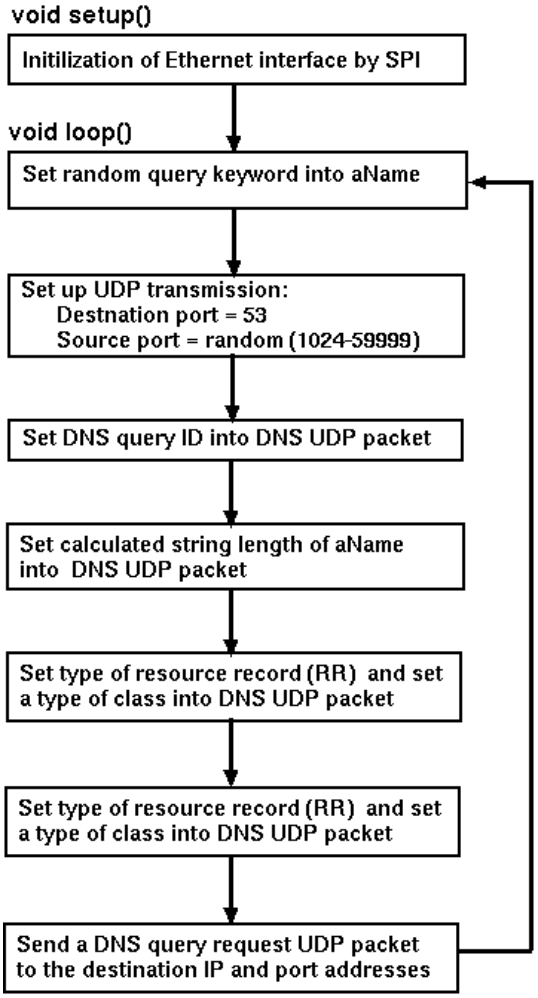

Figure 8 Algorithm in the DNS query request packet traffic generating system (DQRPTGS).

The program source code is as follows:

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <Dns.h>
#include <util.h>
#include <string.h>
#define TYPE_ANY      (0x00FF)
#define CLASS_IN      (0x0001)
#define LABEL_COMPRESSION_MASK (0xC0)
#define DNS_PORT 53
byte mac[]={0x00,0x50,0xC2,0x97,0x28,0x3B};
unsigned int localPort
IPAddress OreServer(192, 168, 10, 1);
IPAddress TargetDNS(192, 168, 10, 1);
IPAddress myIP(192, 168, 10, 10);
IPAddress myDNS(192, 168, 10, 1);
IPAddress myGateway(192, 168, 10, 254);
IPAddress mySubNetmask(255,255,255,0);
const int ORE_PACKET_SIZE= 256;
EthernetUDP Udp;
uint16_t DqID=0;
uint16_t twoByteBuffer;
byte aName[ORE_PACKET_SIZE];
void setup()
{
  DNSClient dns;
  Ethernet.begin(mac, myIP,
    myDNS,   myGateway,
    mySubNetmask);
}
void loop()
{
  randomSeed(millis());
  sprintf((char *)aName,
   "t%07ld.kumamoto-u.ac.jp",
   random(9999999));
  Udp.begin(random(1024,9999));
  Udp.beginPacket(TargetDNS, 53);
  DqID=millis();
  Udp.write((uint8_t*)&DqID, sizeof(DqID));
  twoByteBuffer = htons(QUERY_FLAG
   |OPCODE_STANDARD_QUERY
   | RECURSION_DESIRED_FLAG);
  Udp.write((uint8_t*)&twoByteBuffer,
   sizeof(twoByteBuffer));
  twoByteBuffer = htons(1);
  Udp.write((uint8_t*)&twoByteBuffer,
   sizeof(twoByteBuffer));
  twoByteBuffer = 0;
  Udp.write((uint8_t*)&twoByteBuffer,
   sizeof(twoByteBuffer));
  Udp.write((uint8_t*)&twoByteBuffer,
   sizeof(twoByteBuffer));
  Udp.write((uint8_t*)&twoByteBuffer,
   sizeof(twoByteBuffer));
  byte *start = aName;
  byte *end = start;
  uint8_t len;
  while (*end)   {
    end = start;
    while (*end && (*end != '.')) {end++;}
    if (end-start > 0) {
        len = end-start;
        Udp.write(&len, sizeof(len));
        Udp.write((uint8_t*)start,
          end-start);
    }
    start = end+1;
  }
  len = 0;
  Udp.write(&len, sizeof(len));
  twoByteBuffer = htons(TYPE_ANY);
  Udp.write((uint8_t*)&twoByteBuffer,
   sizeof(twoByteBuffer));
  twoByteBuffer = htons(CLASS_IN);
```

```
    Udp.write((uint8_t*)&twoByteBuffer,
      sizeof(twoByteBuffer));
    Udp.endPacket();
    Udp.stop();
    delay(10);
  }
```

This program can generate 62-63 queries per second. For example, the DNS query request packet generating system (DGRPGS) can totally generate round 250 queries per second, totally.

3. Results and Discussion

3.1 Score of DNS ANY request cannon activity and inbound ANY resource record based DNS query request packet traffic

We illustrate the calculated score of the DNS Query Request Cannon (DARC) activity using restricted Damerau-Lehvenshtein distance based on detection model ($LD(DN_i, DN_{i-1}) = 0$) between the current domain name DN_i and the last domain name DN_{i-1}, as the DNS query keywords in the ANY resource record (RR) based DNS query request packet traffic from the Internet to the top DNS (tDNS) server from January 1st, 2011 to December 31st, 2012, as shown in Figure 9.

In Figure 9, we can observe that the DARC activity score curve takes a zero value and it starts to change drastically since November 28th, 2011. Also, we can observe that the inbound ANY RR based DNS query request packet traffic curve changes in a mild manner before November 28th, 2011. However, both curves change in almost the same manner after November 28th, 2011. This feature indicates that the DARC activity score is significantly correlated with the traffic value of the inbound ANY RR based DNS query request packet access.

In the score curve for the DARC activity, we can observe nine significant peaks (1)-(8), exceeding more than a score value of 35,000day^{-1} and being allocated to (1) May 18th, (2) 29th, (3) June 1st, (4) July 7th, (5) 10th, (6) 16th, (7) August 20th, and (8) October 19th, 2012, respectively. In the peaks, the peak (2) is the top score (=597,977day^{-1}), and we also investigated the source IP address changes in the ANY RR based DNS query request packet traffic through May 29th, 2012, and the results are shown in Figure 11, in order to check whether the source IP addresses are distributed or not.

Interestingly, in Figure 11, we can view scenery

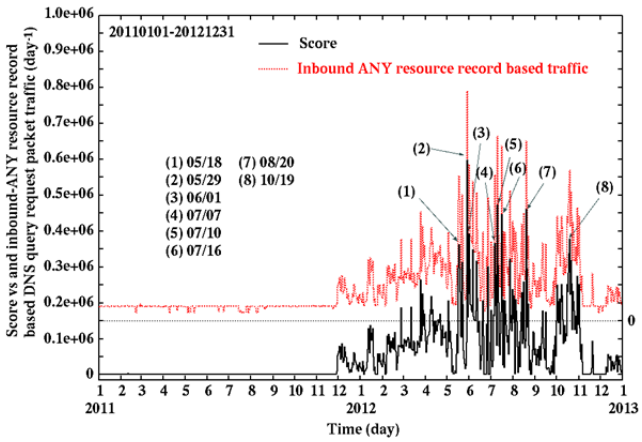

Figure 9 Changes in score of the DNS ANY request cannon (DARC) activity (solid curve) and the inbound ANY resource records (RR) based DNS query request packet traffic to the top DNS (tDNS) server (dotted curve) through January 1st, 2011 to December 31st, 2012 (day^{-1} unit).

```
May 29 23:58:19 kun named[5595]: client **4.**8.**8.*30#61617: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:19 kun named[5595]: client **8.*8.20.*8#54839: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:19 kun named[5595]: client **4.**8.**8.*30#7805: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:20 kun named[5595]: client 2**.**8.**4.**7#54431: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:21 kun named[5595]: client **8.*8.20.*8#64503: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:21 kun named[5595]: client 2**.**8.**4.**7#45380: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:22 kun named[5595]: client 1**.1**.**.*4#47365: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:22 kun named[5595]: client **8.*8.20.*8#62936: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:23 kun named[5595]: client **4.**8.**8.*30#54618: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:24 kun named[5595]: client 2**.**8.**4.**7#32481: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:24 kun named[5595]: client **4.**8.**8.*30#806: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:24 kun named[5595]: client 2**.**8.**4.**7#3926: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:25 kun named[5595]: client **8.*8.20.*8#56393: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:25 kun named[5595]: client **8.*8.20.*8#60067: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:25 kun named[5595]: client 1**.1**.**.*#22420: query: kumamoto-u.ac.jp IN ANY +
May 29 23:58:25 kun named[5595]: client 1**.1**.**.*#3146: query: kumamoto-u.ac.jp IN ANY +
```

Figure 10 Changes in the source IP address in the total ANY-resource records (RR) based DNS query request packet traffic from the Internet to the top domain DNS (tDNS) server at May 29th, 2012.

that the source IP addresses change periodically, showing that the DARC activity is carried out in a source IP address distributed manner.

3.2 Frequency distributions of Euclidian distance in source IP addresses

We calculated frequency distributions of the Euclidian distance for the two peaks (2) May 29th and (5) July 10th, 2012, as shown in Figure 11. In Figure 11, the frequency distribution of the peak (2) has a significant peak at zero value and the other one (10) has also a peak at zero but several peaks take a range of 0.0-350.0 in a broad manner. This feature indicates that there are two types of DARC activities. One is a source IP address non-distributed activity and the other one is a source IP address distributed one.

Interestingly, however, the source IP address distributed activity was also reported in the host

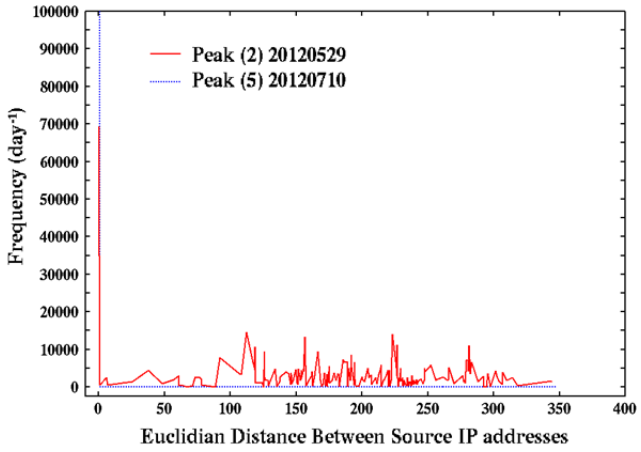

Figure 11 Frequency distributions of the Euclidian distance between the source IP addresses at May 29th and July 10th, 2011 (day^{-1} unit).

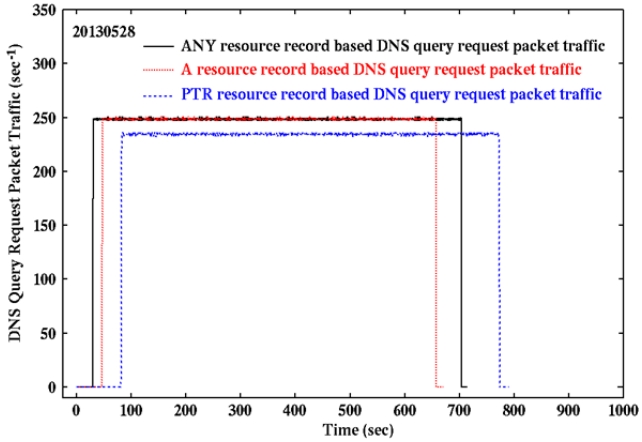

Figure 12 Changes in the ANY, A, and PTR resource records (RRs) based DNS query request packet traffics in May 28th, 2013 (sec^{-1} unit).

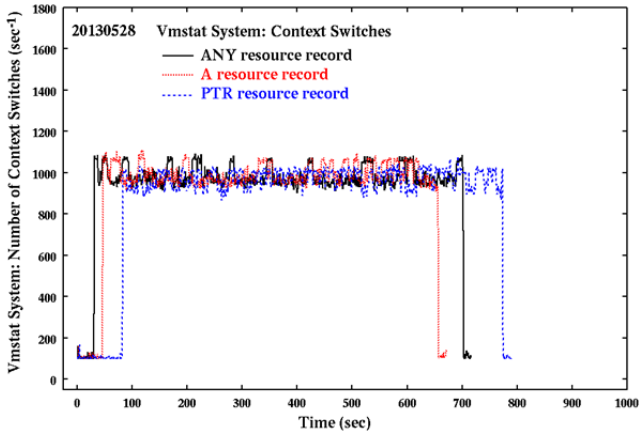

Figure 13 Changes in the number of *context switches* in *vmstat* system parameters at May 28th, 2013 (sec^{-1} unit).

search (HS) activity [12] and the frequency distribution in the HS activity takes a range of 1.0- 5.0 (day^{-1}). This difference shows that the source IP address distributed DARC activity does not resemble well with the source IP with the source IP address distributed HS one. Thus, it can be concluded the frequency distribution is not useful for detection of the DARC activity.

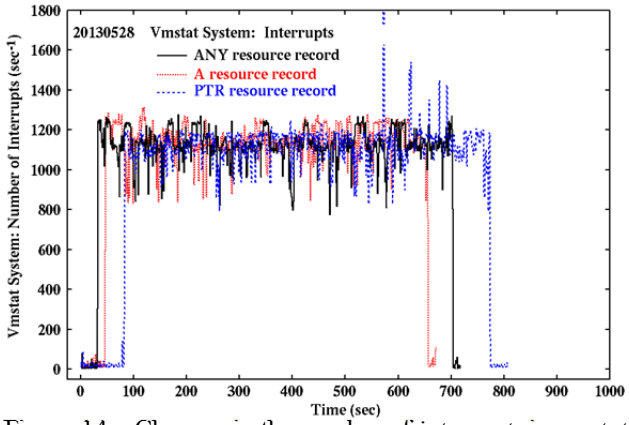

Figure 14 Changes in the number of *interrupts* in *vmstat* system parameters at May 28th, 2013 (sec^{-1} unit).

3.3 DNS query loading tests

We performed the loading test generating the ANY, A, and PTR resource records (RRs) based DNS query request packet traffics to a test DNS server in the campus network, using the DNS query request packet traffic generating system (DQRPTGS), at May 28th, 2013, in which the test server consists of a BIND-9.8.2 DNS server program package [11] in CentOS 6.4 [16] as a guest operating system, an instance in VirtualBox4.1.10 [17], a Scientific Linux 6.0 x86_64 [18] as a host operating system,

As shown in Figure 12, the system generated 250 queries per second (qps) in the ANY and A RRs based DNS query request packet traffics, and 240 qps in the PTR RR based DNS query request packet traffic. These traffics were captured and observed with a network protocol analyzer tool *Wireshark* [19].

In the DNS query request packet traffics, the unique DNS query keywords were generated to avoid to be affected with the DNS resolver cache in the test DNS server.

Through performing the loading test, we also observed two *vmstat* system parameters in the test DNS server, like the numbers of *context switches* and *interrupts*, and we obtained results as shown in Figures 13 and 14, respectively.

Unexpectedly, in Figures 13 and 14, we can observe only small differences among both *vmstat* system parameters corresponding to the ANY, A and PTR resource records based DNS query request packet traffics.

Consequently, it can be concluded that the

DARC activity is almost the same load as the other A or PTR RR based DNS request packet traffic like the conventional DNS cache poisoning attack [8] or the host search (HS) activity [12].

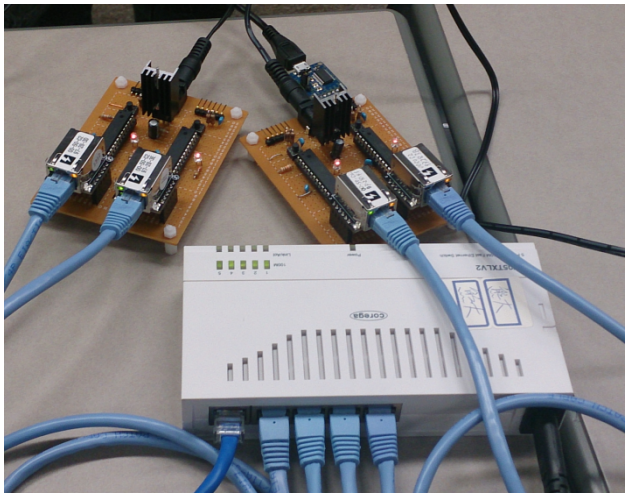

Figure 14. DNS query request packet traffic generating system (DQRPTGS).

4. Conclusions

We developed and evaluated the restricted Damerau-Levenshtein edit distance-based detection model of the DNS any request cannon (DARC) traffic in the inbound ANY resource record (RR) based DNS request packet traffic from January 1st, 2011 to December 31st, 2012.

The following interesting results are found: (1) we observed that the detection score of the DARC traffic was significantly correlated with the inbound ANY RR based DNS query request packet traffic since November 28th, 2011. (2) We observed the source IP addresses are distributed in the DARC traffic. (3) We also carried out the loading test generating the ANY, A, and PTR RRs based DNS queries to the test DNS server and unexpectedly, we found that both changes in *vmstat* parameters were almost the same each other i.e. these features indicate the DARC activity can be almost the same as the conventional Kaminsky (DNS cache poisoning) attack or the host search (HS) activity.

Finally, it can be concluded that the DARC activity should be a pre-investigation activity, a fake activity, and/or a faint activity before cyber attack against the DNS servers or DNS services in the campus network.

Acknowledgments

This work was supported by Japan Society for the Promotion of Science (JSPS) KAKENHI (Grant-in-Aid for Challenging Exploratory Research) Grant Number 12013489.

References

[1] T. Daly, "Observed DNS Anomaly: Bumps in DNS ANY Query Activity," *Dyn Inc*., Manchester, NH, 2011,http://www.dyncommunity.com/questions/22190/observed-dns-anomaly-bumps-in-dns-any-query-activi.html

[2] K. Shortt, "DNS ANY Request Cannon - Need More Packets", *Internet Storm Center (ISC) Diary*, SANS Technology Institute 2012, https://isc.sans.edu/diary.html?date=2012-05-21

[3] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "A Fair Solution to DNS Amplification Attacks", In: *Proc. of the Workshop on Digital Forensics and Incident Analysis 2007 (WDFIA2007)*, Karlovassi, Samos, Greece, pp.38-47, 2007.

[4] M. Prince, "Deep Inside a DNS Amplification DDoS Attack", *ClouFlare, 2012*, http://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack

[5] J. Nazario, "DDoS attack evolution: Computer Security Series", *Network Security*, Vol.2008, No.4, pp.7-10,2008.

[6] V. L. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", Soviet Physics Doklady, Vol. 10, No. 8, pp.707-710, 1966.

[7] F. J. Damerau, "A technique for computer detection and correction of spelling errors", Communications of the ACM, Vol. 7, No. 3, pp.171-176 (1964).

[8] Y. Musashi, M. Kumagai, S. Kubota, and K. Sugitani, "Detection of Kaminsky DNS Cache Poisoning Attack", In: *Proc. Of the Fourth International Conference on Intelligent Networks and Intelligent Systems (ICINIS 2011)*, Kunming, China, pp. 121-124, 2011.

[9] Atmel AVR Atmega328P-PU: http://www.atmel.com/devices/ATMEGA328P.aspx

[10] Wiznet W5100/W5200 Ethernet chip: http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5200_DSV129E.pdf

[11] BIND-9.8.2: http://www.isc.org/products/BIND/

[12] N. Shibata, Y. Musashi, D. A. Ludeña Romaña, S. Kubota, and K. Sugitani, "Trends in Host Search Attack in DNS Query Request Packet Traffic", In: *Proc. of the Fifth International Conference on Intelligent Networks and Intelligent Systems (ICINIS 2012), Tianjin, China, pp.126-129, 2012.*

[13] optiboot: http://code.google.com/p/optiboot/

[14] Wiznet WIZ820io: http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=&cate2=&cate3=&pid=1161

[15] Arduino-IDE ver.1.0.3: http://Arduino.cc/en/main/software

[16] CentOS 6.4: http://www.centos.org/

[17] VirtualBoX4.1.10: https://www.virtualbox.org/

[18] Scientific Linux 6.0: https://www.scientificlinux.org/

[19] Wireshark: http://www.wireshark.org/