# DNS Cache Poisoning Attack Analysis and Detection Using Packet Header

B. B. Jayasingh
Professor, CVR College of Engineering/IT Department, Hyderabad, India
Email : bbjayasingh9@rediffmail.com.

***Abstract*- DNS is the most critical component in the internet and its security is crucially challenged. The normal operation of DNS is to acquire the correct domain to IP address mapping to browse the web and sending emails etc. DNS cache poisoning is an attack strategy that diverts the network traffic to attacker's computer by exploiting the vulnerabilities in the DNS server. The web server is susceptible to many kinds of attacks. One such attack called DNS cache poisoning attack is discussed here. The attack procedure of the DNS cache is briefly narrated so that the algorithmic solution is easily developed by following an IF then Else conditional rules. In this paper, we developed an algorithm to detect DNS cache poisoning attack based on packet header analysis. The DNS packets are captured through ETHEREAL software and stored in a log after dissection of the packet header. The logged packets are analyzed and processed through an algorithm in order to detect the possibility of DNS Cache Poisoning attack.**

***Index Terms*- DNS Cache Poisoning Attack, DNS reply, TTL, Request transaction ID, Reply transaction ID**

## I. Introduction

The most vulnerable layers among the seven layers of OSI reference model are the network layer and the transport layer [1]. There are many attacks on these two layers by the third party attacker either intentionally or due to misconception. We assume that the network is susceptible to many such attacks where a system or server is compromised. These attacks [2] greatly affect the system performance and destroying the valuable information on the web server. It is not easy to detect or prevent some of the attacks due to the intelligence of third party attackers. The attackers violate all the security rules [3] and also remove the trespassing activity on the web from the server logs. Therefore the network traffic has to be monitored and a suitable method has to be developed for countering the attack.

The simple way of monitoring the network traffic is to look at the server log files. The log files contain the activities related to their trespassing i.e. date, time, IP address, HTTP status, bytes sent, bytes received etc. The log files are the most important documents to the security professionals where the histories of certain attacks are found. In this paper, we also look at the log files at server for the entries of DNS traffic and devise an algorithm to detect DNS Cache Poisoning attack.

## II. Background

The domain names are converted to IP addresses in the case of DNS and it is the bridge between the client and the server. There is an analytical model which defines the DNS infrastructure along with the selection of DNS parameters [4]. Similarly, they have compared the performances of DNS security solutions using cryptographically approach and collaborative overlay approach.

The DNSSEC (DNS SECurity) is essential for DNS cache poisoning attack as the attacker redirects the communications to adversarial servers. DNS cache poisoning attack indirectly allows interception of original contents and modification [5].

It is a considerable threat to the security of internet users which has to be detected at the earliest. Some such detection methods are implemented by using Kalman filter and the entropy of query packet [6]. Another mechanism called security proxy is also implemented, the schemes to prevent the DNS cache poisoning attack are the selective requery and the security level communication [7]. There is a novel solution to DNS cache poisoning attacks called WSEC DNS (Wild card SECure DNS) which depends on existing DNS protocol but with a wild card domain name.

A system or server in a network is susceptible to many attacks. The attacks [8] can be of different types which may include the attacks done intentionally by the third party. Firewall does not protect the system from such attacks. Some of the existing software and techniques are discussed hereunder.

- *Basic Entry Log Auditing Software*

A Basic Entry system allows you or a log auditor to enter the logs manually onto the system and the software indicates if there is Hours of Service (HOS) violation and so forth. The auditor will be allowed to deduct points off the driver if he has for example; not calculated his hours correctly, missing remarks, etc. From there you have a reporting package that will highlight all the drivers' mistakes and give him a rating percentage for any violation that he/she has done.

DOI:10.32377/cvrjst1218

The software comes with one rule set (Canadian or American), Manual Event Tracking, Initial 50 Active Drivers and a Reporting package.

- *Automated Scanning & Log Auditing Software*

An automated system will take a scan able log and do the auditing for you. The software will take all the information from the drivers log and red flag any logs with violations. If 100 logs were to be scanned in to the system and 20 of the logs had violations on them. You will only be auditing the 20 logs that were red flagged.

The software comes with Driver & Vehicle Profiles, Timed Events Verification Module, Full Reporting Package, Initial 50 Active Drivers and Four hours of internet based installation and training assistance.

- *Electronic Log Auditing*

Laptop application with complete rule sets and voice prompts (alerting the driver of pending violations before they happen, based on each driver's individual duty cycle). E-logs will reduce audit review and help enhance communication with corporate offices. It also eliminates paper logs, log editing and HOS violations giving the company and the driver peace of mind to.

- *Network Traffic Monitor Analysis*

Packet Analyzer enterprise edition is an advanced network monitoring, analysis and reporting tool. It captures and analyzes traffic in realtime. It also presents comprehensive and graphic reports for many technical and business applications. All information is displayed in simple English with easy to use interface for anyone to master the tool with minutes of self training.

- *PC-Based Data Logging and Recording Techniques*

Data logging and recording is a very common measurement application. In its most basic form, data logging is the measurement and recording of physical or electrical parameters over a period of time. The data can be temperature, strain, displacement, flow, pressure, voltage, current, resistance, power, or any of a wide range of other parameters. Real-world data logging applications are typically more involved than just acquiring and recording signals, typically involving some combination of online analysis, offline analysis, display, report generation, and data sharing. Moreover, several data logging applications are beginning to require the acquisition and storage of other types of data, such as recording sound and video in conjunction with the other parameters measured during an automobile crash test.

Data logging is used in a broad spectrum of applications. Chemists record data such as temperature, pH, and pressure when performing experiments in a lab. Design engineers log performance parameters such as vibration, temperature, and battery level to evaluate product designs. Civil engineers record strain and load on bridges over time to evaluate safety. Geologists use data logging to determine mineral formations when drilling for oil. Breweries log the conditions of their storage and brewing facilities to maintain quality.

- *Google security tool uses Google to scan sites for vulnerabilities*

Google Scanner is a Web auditing tool released by the hacker group Cult of Dead Cows. The tool uses the prowess of the search engine to surface vulnerabilities on Web sites.

## III. DNS Cache Poisoning Attack

Normally a computer uses DNS (Domain Name System) server [9] to find the domain name of a site. The DNS server responds with one or more IP addresses where a computer can reach to the site. Afterwards a computer connects directly to that numerical IP address. The role of DNS is to convert human-readable addresses like "google.com" to computer readable IP addresses like "173.194.67.102".

DNS cache poisoning [10] is an attack strategy that diverts the network traffic to attacker's computer by exploiting the vulnerabilities in the DNS server. It is also called as DNS Spoofing which corrupts DNS data and returns an incorrect IP address to a computer. Whenever a computer asks for an IP address of a domain name, if the server does not know then it will ask another server and the process continues until it finds. To increase performance the server cache these translations for certain amount of time to make use in future. Now if the server receives same kind of request then it can reply from the cache without asking other servers until the cache expires. But what if a computer receives a false translation and caches for performance optimization then it is considered as poisoned. As the DNS server is poisoned, it replies an incorrect IP address that diverts the network traffic to attacker's computer.

An attacker takes advantage of the loop holes of the DNS software by poisoning the cache and makes it accept incorrect information. If the DNS server does not properly validate the DNS response from an authoritative source then the server will cache the incorrect entries. Now server will reply to users the same incorrect entries for the same kind of request. In this process the attackers get the diverted network traffic and may control the users.

DOI:10.32377/cvrjst1218

To perform a cache poisoning attack [11], the attacker exploits a flaw in the DNS (Domain Name Server) software that can make it accept incorrect information. If the server does not correctly validate DNS responses to ensure that they have come from an authoritative source, the server will end up caching the incorrect entries locally and serve them to users that make the same request.

This technique can be used to replace arbitrary content for a set of victims with content of an attacker's choice. For example, an attacker poisons the IP address in the DNS entries for a target website on a given DNS server, replacing them with the IP address of a server he controls. He then creates fake entries for files on the server they control with names matching those on the target server. These files could contain malicious content, such as a worm or a virus. A user whose computer has referenced the poisoned DNS server would be tricked into thinking that the content comes from the target server and unknowingly downloads malicious content.

We studied the behavior of the DNS Cache Poisoning attack and formulated the algorithm based on packet header to detect the threat. The information about the attack is maintained in a log for the evidentiary purpose in the court of law. Hence the work is embodied with the appropriate results.

### *Attack Procedure*

1. Host requests message to DNS with domain name of destination
2. DNS redirects the request message to sub domains.
3. Attacker captures the request message and finds the transaction ID.
4. Attacker replies with same transaction ID, his own IP address and TTL is set to a greater value.
5. When original reply comes, DNS sees this and thinks that this is a Duplicate and discards it.
6. DNS attack is successful.
7. End.

## IV. Analysis Of DNS Packet

The DNS packet is composed of four important header formats such as MAC header, IP header, UDP header and DNS header as shown in table 1. The details of the header with respect to the DNS request are shown in table 2 and the DNS reply is shown in table 3.

TABLE 1.
THE DNS PACKET FORMAT

| | | | |
|---|---|---|---|
| MAC Header | IP Header | UDP Header | DNS Section |

TABLE 2.
DNS REQUEST PACKET FORMAT

| | | | | |
|---|---|---|---|---|
| Identification (Transaction ID) | Parameters (Flags) | No. of questions | No. of answers | No. of authority |
| No. of additional | Query domain name | Query type | Query class | |

TABLE 3.
DNS REPLY PACKET FORMAT

| | | | | |
|---|---|---|---|---|
| Identification (Transaction ID) | Parameters (Flags) | No. of questions | No. of answers | No. of authority |
| No. of additional | DNS Query section | Answers section | Authoritative Name servers | Additional records |

### *A. DNS Request packet*

We have captured the DNS request packet using the Ethereal software and the details of the packet dissection are given below.

HEX packet format:

```
0000  00 11 85 c2 82 e6 00 16  76 89 b2 58 08 00 45 00
0010  00 41 19 31 00 00 80 11  50 b5 c0 a8 14 89 cb 99
0020  2f fb 04 39 00 35 00 2d  70 c9 ab cd 01 00 00 01
0030  00 00 00 00 00 00 04 61  75 74 6f 06 73 65 61 72
0040  63 68 03 6d 73 6e 03 63  6f 6d 00 00 01 00 01
```

Ethernet II:

- 6 bytes destination address: 00: 11: 85: c2: 82: e6
- 6 bytes source address: 00: 16: 76: 89: b2: 58(192.168.20.137)
- Type: IP (0X0800)08 00

Internet protocol:

- The IP version is IPV4: 4 (45)
- 1 byte differentiated services field: 0x00 (00)
- 2 bytes of total length: 65 (00 41)
- 2 bytes identification: 0x1931 (19 31)
- 1 byte for flags: 0x00 (00)
- 2 bytes of fragment offset: 0 (00 00)
- 1 byte is for time to live:128 (80)
- 1 byte to indicate the protocol: udp (0x11) (11)
- 2 bytes for header checksum: 0x50b5 (50 b5)
- 4 bytes used for source address: 192.168.20.137 (c0 a8 14 89)

DOI:10.32377/cvrjst1218

- 4 bytes for destination address: 203:153:47:251 (cb 99 2f fb)
- User datagram protocol:
- Source port is of 2 bytes: 1081 (04 39)
- Destination port is of 2 bytes: domain (53) (00 35)
- Length: 45 (00 2d)
- Checksum: 0x70c9 (70 c9)

Domain Name System: Query

- The transaction id is 2 bytes: 0xabcd (abcd)
- Flags is of 2 bytes: 0x0100 (01 00)
- Questions is 2 bytes: 1 (00 01)
- Answers is 2 bytes R Rs: 0 (00 00)
- Authority is of 2 bytes: 0 (00 00)
- Additional: 0 (00 00)

Queries:

- Name: auto.search.msn.com (04 61 75 74 6f 06 73 65 61 72 63 68 03 6d 73 6e 03 63 6f 6d 00)
- Type: host address (00 01)
- Class: inet (00 01)

## IV. Implementation

We capture the DNS packets using Ethereal software. After capturing the packets, we store the packets in the database. We store both the request as well as response packets in the database. The request transaction ID is compared with the response transaction ID, if both are equal then check the TTL field in the response answer section. If TTL field is greater than 1 day, then we conclude that there is a DNS Cache poisoning attack.

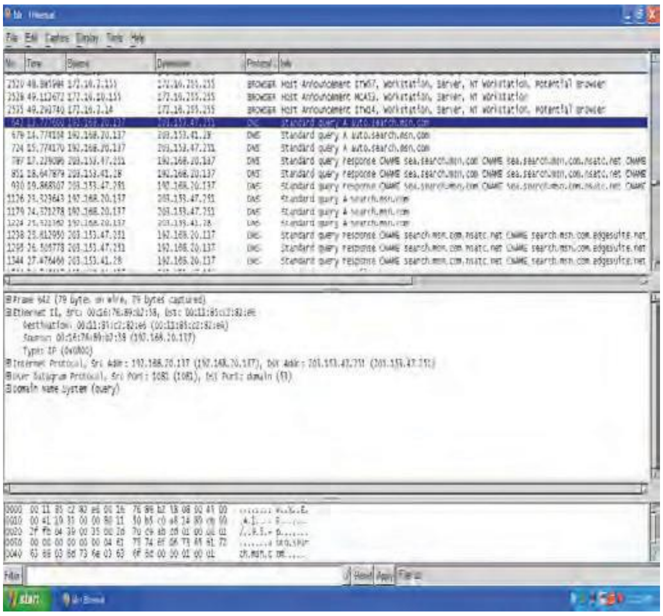
### *A. DNS_Cache_Poisoning Detection Algorithm*

[DNS_DB refers to database of DNS traffic, PKT refers to a DNS packet, DNS_REQ refers to request packet of DNS cache traffic, DNS_REP refers to reply packet of DNS cache traffic, REQ_TR_ID refers to transaction ID of request packet, REQ_IP_ADD refers to IP address of request packet, REQ_DOM_NAME refers to domain name request packet, REP_TR_ID refers to transaction ID reply packet, REP_TTL refers to TTL in the answer section of reply packet, REP_IP_ADD refers to IP address of reply packet.]

```
1.  Capture all packets
    {
    FILTER DNS cache packets;
    STORE in DNS_DB;
    }
2.  If (PKT=DNS_REQ)
3.  {
4.  GET REQ_TR_ID, REQ_IP_ADD, REQ_DOM_NAME;
5.  }.
6.  If (PKT=DNS_REP)
7.  {
8.  GET REP_TR_ID, REP_TTL, REP_IP_ADD;
9.  }
10. If REQ_TR_ID = REP_TR_ID then
    {
      If Ans_TTL >1 day Then
    {
    DETECT as "DNS Cache poisoning attack"
    DISCARD the packet
    STORE the REQ_IP_ADD, REP_TTL, REQ_TR_ID, DATE, TIME in DNS_FORENSIC_LOG.
    }
    }
    ELSE
{
   If REP_TTL <1 DAY Then
    SEND REPLY
}
11. End
```

### *B. Results*

We have captured the DNS packets by using the freeware ETHEREAL Network Analyzer and the details are shown in fig. 1.

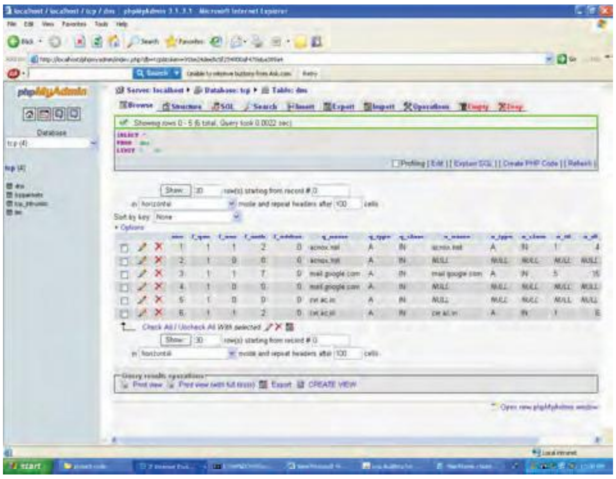


```
No.  Time      Source          Destination     Protocol Info
2520 48.845944 172.16.2.155    172.16.255.255  BROWSER  Host Announcement ITW67, Workstation, Server, NT Workstation, Potential Browser
2526 49.112672 172.16.20.155   172.16.255.255  BROWSER  Host Announcement MCA53, Workstation, Server, NT Workstation
2535 49.290740 172.16.2.14     172.16.255.255  BROWSER  Host Announcement ITW04, Workstation, Server, NT Workstation, Potential Browser
 642 13.777600 192.168.20.137  203.153.47.251  DNS      Standard query A auto.search.msn.com
 679 14.774104 192.168.20.137  203.153.41.28   DNS      Standard query A auto.search.msn.com
 724 15.774170 192.168.20.137  203.153.47.251  DNS      Standard query A auto.search.msn.com
 787 17.229095 203.153.47.251  192.168.20.137  DNS      Standard query response CNAME sea.search.msn.com CNAME sea.search.msn.com.nsatc.net CNAME
 851 18.647879 203.153.41.28   192.168.20.137  DNS      Standard query response CNAME sea.search.msn.com CNAME sea.search.msn.com.nsatc.net CNAME
 930 19.868507 203.153.47.251  192.168.20.137  DNS      Standard query response CNAME sea.search.msn.com CNAME sea.search.msn.com.nsatc.net CNAME
1126 23.323643 192.168.20.137  203.153.47.251  DNS      Standard query A search.msn.com
1179 24.371278 192.168.20.137  203.153.47.251  DNS      Standard query A search.msn.com
1224 25.521362 192.168.20.137  203.153.41.28   DNS      Standard query A search.msn.com
1238 25.612950 203.153.47.251  192.168.20.137  DNS      Standard query response CNAME search.msn.com.nsatc.net CNAME search.msn.com.edgesuite.net
1295 26.505778 203.153.47.251  192.168.20.137  DNS      Standard query response CNAME search.msn.com.nsatc.net CNAME search.msn.com.edgesuite.net
1344 27.476460 203.153.41.28   192.168.20.137  DNS      Standard query response CNAME search.msn.com.nsatc.net CNAME search.msn.com.edgesuite.net

⊞Frame 642 (79 bytes on wire, 79 bytes captured)
⊟Ethernet II, Src: 00:16:76:89:b2:38, Dst: 00:11:85:c2:82:e6
    Destination: 00:11:85:c2:82:e6 (00:11:85:c2:82:e6)
    Source: 00:16:76:89:b2:38 (192.168.20.137)
    Type: IP (0x0800)
⊞Internet Protocol, Src Addr: 192.168.20.137 (192.168.20.137), Dst Addr: 203.153.47.251 (203.153.47.251)
⊞User Datagram Protocol, Src Port: 1081 (1081), Dst Port: domain (53)
⊞Domain Name System (query)

0000  00 11 85 c2 82 e6 00 16  76 89 b2 38 08 00 45 00   ........v..8..E.
0010  00 41 19 31 00 00 80 11  50 b5 c0 a8 14 89 cb 99   .A.1....P.......
0020  2f fb 04 39 00 35 00 2d  70 c9 ab cd 01 00 00 01   /..9.5.-p.......
0030  00 00 00 00 00 00 04 61  75 74 6f 06 73 65 61 72   .......a uto.sear
0040  63 68 03 6d 73 6e 03 63  6f 6d 00 00 01 00 01      ch.msn.c om.....
```

Figure1. Frame which captures the DNS packets

DOI:10.32377/cvrjst1218

The DNS packets are dissected and the required header fields are stored in the MySQL database for further analysis of attack and detection as shown in fig. 2.



Server: localhost ▸ Database: tcp ▸ Table: dns

Showing rows 0 - 5 (6 total, Query took 0.0022 sec)

Show: 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Sort by key: None

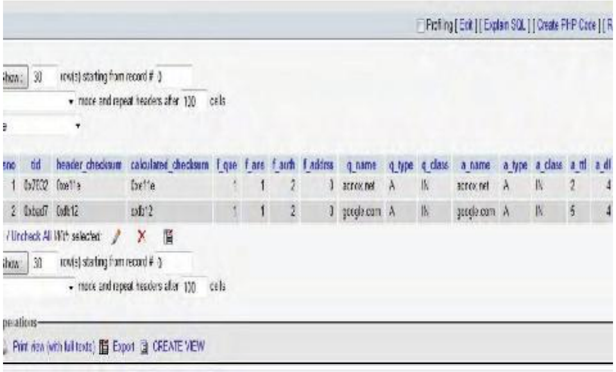
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 0 | acnox.net | A | IN | acnox.net | A | IN | 1 | 4 |
| 2 | 1 | 0 | 0 | 0 | acnox.net | A | IN | NULL | NULL | NULL | NULL | NULL |
| 3 | 1 | 1 | 7 | 0 | mail.google.com | A | IN | mail.google.com | A | IN | 5 | 15 |
| 4 | 1 | 0 | 0 | 0 | mail.google.com | A | IN | NULL | NULL | NULL | NULL | NULL |
| 5 | 1 | 0 | 0 | 0 | cvr.ac.in | A | IN | NULL | NULL | NULL | NULL | NULL |
| 6 | 1 | 1 | 2 | 0 | cvr.ac.in | A | IN | cvr.ac.in | A | IN | 1 | 8 |

Check All / Uncheck All With selected:

Query results operations: Print view | Print view (with full texts) | Export | CREATE VIEW

Figure 2. Stored packets in MySQL database

The detected intrusion packets are stored in the DNS_INTRUSION database for the purpose of future evidentiary purpose by the law enforcement agencies as shown in fig. 3.



Show: 30 row(s) starting from record # 0

mode and repeat headers after 100 cells

| sno | tid | header_checksum | calculated_checksum | f_que | f_ans | f_auth | f_addtnl | q_name | q_type | q_class | a_name | a_type | a_class | a_ttl | a_dl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0x7EC2 | 0xe11e | 0xe11e | 1 | 1 | 2 | 0 | acnox.net | A | IN | acnox.net | A | IN | 2 | 4 |
| 2 | 0xbad7 | 0xdf12 | 0xdf12 | 1 | 1 | 2 | 0 | google.com | A | IN | google.com | A | IN | 5 | 4 |

Uncheck All With selected:

Print view (with full texts) | Export | CREATE VIEW

Figure 3. DNS intrusion database

## V. Conclusions

It is observed that the network traffic entering into the system must be carefully analyzed. With the increase in emerging technologies, there has always been a scope for hackers to trespass into the others system in an innovative way. Therefore in this paper, we have effectively detected the DNS cache poisoning attack and these attacks affect the integrity of the data, and the authentication of the user. Though there are several ways to study or analyze a packet, we study the header format and perform the actions accordingly. Research is also being done in detecting the errors by performing correlation analysis and discriminating the packets.

## References

[1] S. Haber, W.S. Stornetta, How To Time Stamp A Digital Document, Advances In Cryptology, Crypto 90, Springer Verlag, 1999, pp.437-455.

[2] Przemyslaw Kazienko & Piotr Dorosz, "Intrusion Detection Systems (IDS) Part 2 - Classification; methods; techniques, windows security.com, Jun 15, 2004.

[3] Stefano Agnelli, Vic Dewhurst, LAN Interconnection Via ATM Satellite Links For CAD Applications: The UNOM Experiment, In Proceedings Of IEEE ICC, June 1996.

[4] L. Yuan, K. Kant, P. Mohapatra, C.-N. Chuah, "A Proxy View of Quality of Domain Name Service", INFOCOM 2007, 26th IEEE International Conference on Computer Communications, 6-12 May 2007.

[5] Amir Herzberg ; Haya Shulman, "DNSSEC: Security and availability challenges", 2013 IEEE Conference on Communications and Network Security (CNS), 14-16 Oct. 2013.

[6] Hao Wu ; Xianglei Dang ; Liang Zhang ; Lidong Wang, "Kalman filter based DNS cache poisoning attack detection", 2015 IEEE International Conference on Automation Science and Engineering (CASE), 24-28 Aug. 2015.

[7] Lejun Fan ; Yuanzhuo Wang ; Xueqi Cheng ; Jinming Li, "Prevent DNS Cache Poisoning Using Security Proxy", 2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), 20-22 Oct. 2011.

[8] James Figueroa, Brandi Ortega, Shaking Up the Cybersecurity Landscape, IEEE Security and Privacy, Nov/Dec. 2008.

[9] Ge Zhang, Sven Ehlert , Thomas Magedanz, Dorgham Sisalem, Denial of service attack and prevention on SIP VoIP infrastructures using DNS flooding, in Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications, 2007.

[10] Yong Wan Ju Kwan Ho Song Eung Jae Lee Yong Tae Shin, DNS Cache Poisoning Detection Method for Improving Security of Recursive, in proceedings of the $9^{th}$ International Conference on Advanced Communication Technology, 12-14 Feb. 2007,Volume: 3, pp. 1961-1965.

[11] Lihua Yuan Kant, K. Mohapatra, P. Chen-Nee Chuah, DoX: A Peer-to-Peer Antidote for DNS Cache Poisoning Attacks, IEEE International Conference on Communications (ICC '06), Volume: 5, June 2006.

DOI:10.32377/cvrjst1218