International Journal of Innovative Computing, Information and Control
Volume **13**, Number **4**, August **2017**
ICIC International ©2017 ISSN 1349-4198
pp. 1425–1432

DNSAM: A DNS DATA REAL-TIME ANALYSIS AND MONITORING SYSTEM

Ying Liu, Ting Zhi and Zehui Liu

National Engineering Laboratory for Next Generation Internet Interconnection Devices
Beijing Jiaotong University
No. 3, Shangyuancun, Haidian District, Beijing 100044, P. R. China
{ yliu; 15111048; 12111019 }@bjtu.edu.cn

Received November 2016; revised March 2017

Abstract. *Web applications are based on DNS (Domain Name System). Aiming at defending against Internet threats (i.e., phishing, malicious code, etc.) in the Web applications, this paper presents a DNS data real-time analysis and monitoring system. The proposed system is called DNSAM (Domain Name System Analysis and Monitoring) as it uniquely introduces the improved multi-pattern MWM (Modified Wu_Manber) domain name matching algorithm and the non-collision hashing multi-bit Trie-tree IP matching algorithm. By means of quick match between the critical information in DNS packets and DNS security policy, DNSAM successfully completes the suspected information storage and domain name redirecting. Our analysis and the corresponding experimental results show that DNSAM offers an effective performance and a favorable success rate to analyze and monitor the DNS data.*

Keywords: DNS monitoring, Improved MWM matching algorithm, Multi-bit Trie-tree matching algorithm, Domain name redirecting

1. Introduction.

With the rapid development of network technology in recent years, Web applications (e.g., WWW) have been becoming a powerful platform on the Internet. DNS (Domain Name System) is a hierarchical distributed system providing the necessary mapping or binding between human comprehensible domain names and the corresponding numerical IP addresses, and it is the basis of the Web applications. Therefore, as an important network service, its query and response packets include significant information – the mapping relations between Web domain names and IP addresses, which plays a key role for the users to access a Web server.

However, with the continuous development and the widespread use of Web applications, they are facing more and more security issues [1,2], such as phishing, malicious code, spamming, pornography, and Internet gambling. As we know, these illegal behaviors have seriously affected the robust development of Internet environment, but there is no fundamental solution to Web security problems yet.

This paper summarizes current Web security problems, and presents a DNS data real-time analysis and monitoring system – DNSAM, which acts as a third-party to capture and analyze the DNS query and response packets which are transmitting on the Internet [3]. DNSAM uniquely introduces the improved multi-pattern MWM (Modified Wu_Manber) [4,5] domain name matching algorithm and the non-collision hashing multi-bit Trie-tree [6,7] IP matching algorithm, and quickly matches the critical information in DNS query and response packets with the DNS security policy. Compared to the previous algorithms, the proposed algorithms can simultaneously search multiple patterns and support a large set of patterns. Also, the improved algorithms can complete the fast match and have great convergence property and convergence efficiency. At the same time, DNSAM also can redirect the suspicious domain name, which prevents users from accessing malicious Web servers and provides users with legitimate Web servers. The final purpose of DNSAM is to reduce network security threats and facilitate the healthy development of Internet.

The remainder of this paper is organized as follows. Section 2 briefly outlines related researches about current DNS and Web security. Section 3 presents the overall architecture of DNSAM and its functional modules. Section 4 gives the key technologies of this system. Section 5 carries out experiments on this system and gives the performance analysis. Section 6 concludes this paper.

2. **Related Work.** DNSSEC (DNS Security Extensions) [8] is a suite of security extensions to DNS, provided by IETF's DNSSEC Working Group, and it introduces a mechanism for users to verify the origin authenticity and integrity of DNS data based on cryptographic signatures. Although DNSSEC can prevent some illegal behaviors (i.e., phishing, spamming, Internet gambling, etc.) to a certain extent, it has not been popularized and applied widely owing to some problems about the system efficiency and key management.

Dnstop [9] and DSC [10] can carry out the statistical analysis by the use of DNS logs, and identify the abnormal DNS query information in certain time. However, they are passive analysis tools which lack for the real-time.

Independent third-party detection measures (e.g., email detection [11], web page anomaly detection [12], and web page similarity detection [13]) can avoid some illegal behaviors, such as phishing, spamming, Internet gambling. Whereas these measures need to mine and analyze the content and structure of Web pages, and the mathematical characteristics of detection algorithms are very complex and high computational, it is difficult to apply these measures in Internet.

3. **DNSAM Architecture.** As a third-party network monitoring system, DNSAM plays a major role in the access router to monitor the local network, such as campus network, and corporate network. As shown in Figure 1, by quickly capturing and analyzing the DNS packets, DNSAM provides a complete real-time detection scheme.

According to the characteristics of TCP/IP architecture and the time order of the packet processing, DNSAM introduces the hierarchical structure and building block design. It mainly consists of three modules: data capture and analysis module, DNS security policy matching module and application management module. Besides, in order to sufficiently utilize the CPU resources, this system introduces the multi-threading mechanism based on CPU efficiency balancingas shown in Figure 2.

Data Capture and Analysis Module: This module primarily introduces the NAPI mechanism to achieve high-speed packet capture from the Internet, and then filters out all DNS

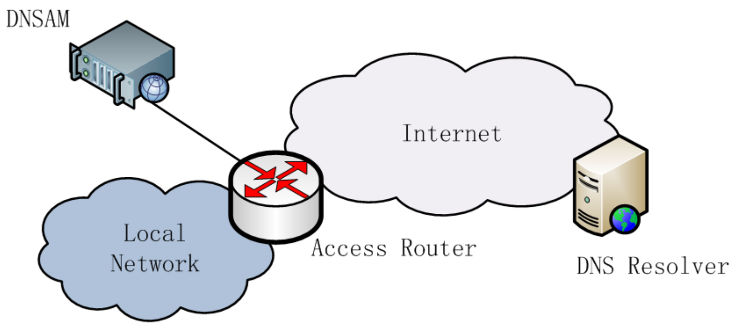

FIGURE 1. Topology of DNSAM

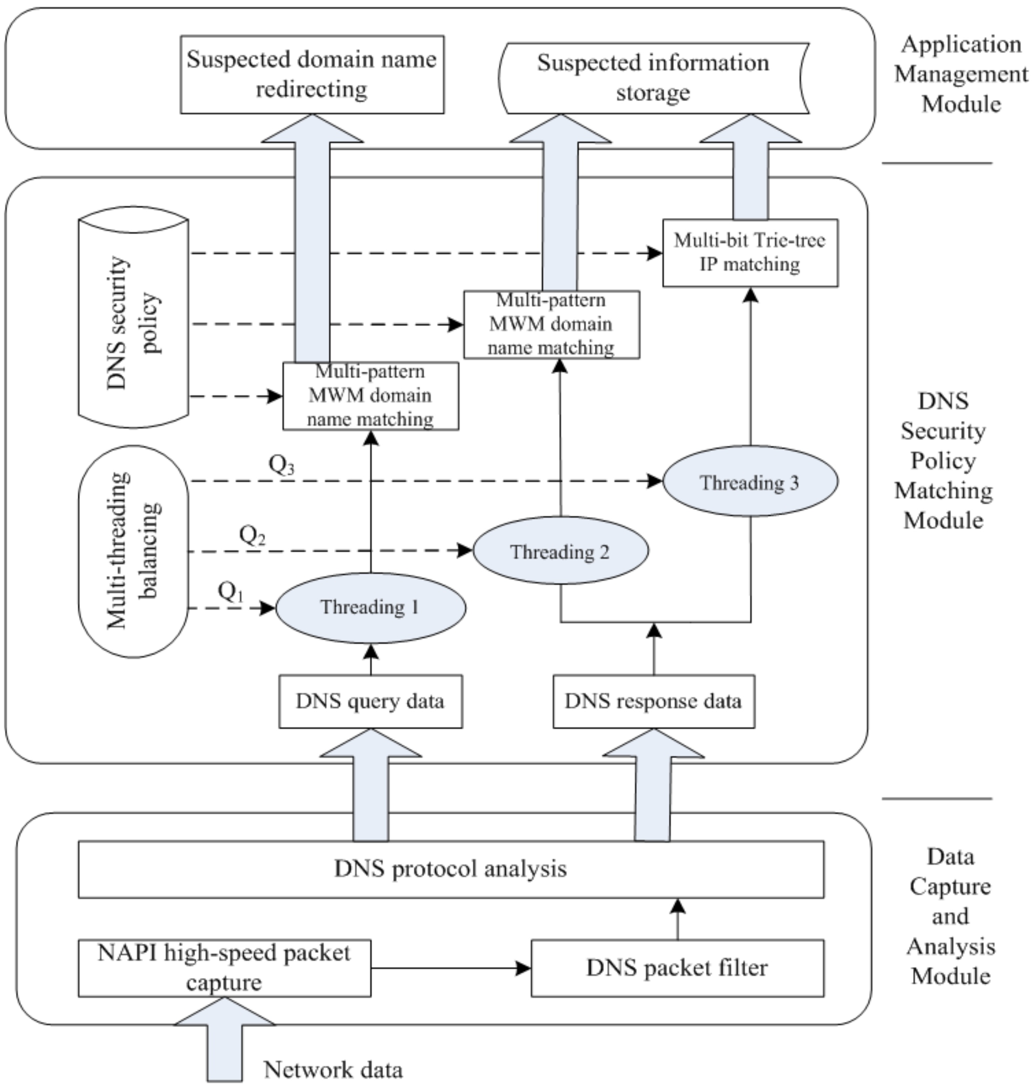
Suspected domain name redirecting
Suspected information storage
Application Management Module
Multi-bit Trie-tree IP matching
Multi-pattern MWM domain name matching
Multi-pattern MWM domain name matching
DNS security policy
DNS Security Policy Matching Module
Multi-threading balancing
Q_3
Q_2
Q_1
Threading 3
Threading 2
Threading 1
DNS query data
DNS response data
DNS protocol analysis
NAPI high-speed packet capture
DNS packet filter
Data Capture and Analysis Module
Network data

FIGURE 2. System architecture of DNSAM

query and response packets. After analyzing these packets, this module educes the critical information (e.g., IP address, and domain name) which will be matched with DNS security policy in DNS Security Policy Matching Module.

DNS Security Policy Matching Module: This is the core of DNSAM. In order to sufficiently utilize the CPU resources, this module adopts the multi-threading mechanism based on CPU efficiency balancing. At the same time, it uniquely introduces the improved multi-pattern MWM (Modified Wu_Manber) domain name matching algorithm and the non-collision hashing multi-bit Trie-tree IP matching algorithm, and completes the quick match between the critical information and DNS security policy. Furthermore, DNS security policy consists of three blacklists: response domain name blacklist, response IP blacklist and domain name redirecting blacklist. The first and second blacklists basically include the illegal or abnormal domain names and IP addresses, and the last blacklist mainly includes the redirecting domain names.

Application Management Module: This module consists of two parts. One is the suspected information storage, which results from the matching outcome between the DNS response packets and the response domain name or IP blacklist. What is more, the suspected information, including time, source and destination IP addresses, suspected domain names, IP addresses corresponding to the suspected domain names and so on, can track the behaviors of users and acquire the computer crime forensics. The other is the suspected domain name redirecting, which results from the matching outcome between the DNS query packets and the domain name redirecting blacklist. Besides, in order to prevent users from accessing malicious Web servers, this operation provides users with secure IP addresses. Accordingly, it safeguards the legitimate rights of Internet users.

4. **Key Technologies.**

4.1. **Improved multi-pattern MWM domain name matching algorithm.**

The improved multi-pattern MWM domain name matching algorithm completes the fast domain name match by comparing the domain names in DNS query packets or DNS response packets with domain name redirecting blacklist or response domain name blacklist. This algorithm can simultaneously search multiple patterns and support a large set of patterns. Furthermore, it introduces the secondary hash to resolve the hash collision problem in the MWM algorithm [4,5]. Because a great number of strings compose the abnormal domain name in domain name redirecting blacklist and response domain name blacklist, the improved multi-pattern MWM domain name matching algorithm can significantly enhance the efficiency of DNSAM. The improved algorithm is divided into two stages: the preprocessing stage and the domain name matching stage. And the detailed steps of this algorithm are as follows.

(1) The preprocessing stage

Step 1: Calculate the shortest domain name string's length in the blacklist, denoted by m.

Step 2: Sort the domain name strings in alphabetical order, and store them in an array, denoted by *PatArray*[]. And then number these strings consecutively from 0, regarded as its identifier.

Step 3: Create a hash table *HASH*, a hash collision statistical table *NumArray* and a secondary hash table *SecHASH*. For each substring which is the composition of the first B byte (actually $B = 2$) of each domain name, the preprocessing stage regards its hash value *index* as the hash address of this domain name, namely *HASH*[*index*] $= i$, where i is its identifier. If more than one domain name string holds the same hash value *index*, it needs to count these domain name strings, denoted by *ngroup*. *ngroup* will be stored in the relevant location of the first domain name string whose hash value is *index*, namely $NumArray[i]$ = *ngroup*. After that, the preprocessing stage calculates the secondary hash value of the multiple domain name strings which have the same *index*, and the secondary hash objects are the shortest prefix characters which can differentiate these domain name strings. Finally, the corresponding secondary hash value will be stored in *SecHASH*.

Step 4: A shift table *SHIFT* will be established by the preprocessing stage, and all the initial values will be set to $m - B + 1$. After that, the preprocessing stage will traverse all of the domain name strings and analyze the first m characters of each string. Firstly, the preprocessing stage achieves the hash value *index* of every substring of size B in turn from left to right. Secondly, it considers the location q of the substring's last character in the m characters. At last, the preprocessing stage will compare $m - q$ with $SHIFT[index]$, if $m - q$ is smaller, then $SHIFT[index]$ will be set $m - q$; and if $m - q$ is bigger, then $SHIFT[index]$ will be unchanged.

(2) The domain name matching stage

Step 1: Calculate the hash value *index* of the last B characters which belong to the current m characters of the domain name information in the DNS packet.

Step 2: Look up $SHIFT[index]$ in the shift table, if $SHIFT[index] > 0$, then shift $SHIFT[index]$ characters to the right and go to Step 1; otherwise, go to Step 3.

Step 3: Shift m characters to the left from the current position, and calculate the hash value of B characters, denoted by *index1*. Afterwards, search the table *HASH* by the use of *index1* and get i'. If $NumArray[i'] = 1$, then compare $PatArray[index1]$ with the corresponding domain name information in the DNS packet; and if $NumArray[i'] > 1$, then implement the secondary hash of the corresponding domain name information in the DNS packet, and then compare this secondary hash value with the table *SecHASH*. At last, find the matching domain name string and go to Step 4. If there are no matching with the table *SecHASH*, and then go to Step 4.

Step 4: Shift one character to the right in the domain name information, and go to Step 1.

4.2. **Multi-bit Trie-tree IP matching algorithm.** The multi-bit Trie-tree IP matching algorithm generates IP blacklist Trie tree by the 16-bit index values that are the non-collision hash values of IP addresses in response IP blacklist, and completes the fast match by comparing the IP addresses in DNS response packets with response IP blacklist. In this algorithm, each branch of IP black Trie tree will point to the suspected information storage operation in application manage module. The process of this algorithm is shown in Figure 3.

The algorithm of IP blacklist Trie tree generates the 16-bit index value by hashing each 32-bit IP address in the response IP blacklist, and uniformly maps these IP addresses to the 16-bit interval. In general, each IP address owns an index value, and all of these index values compose an index table. At last, the algorithm constitutes the IP blacklist Trie tree by the use of the index table, as shown in Figure 4. $f(IP_a, IP_b)$ is the hash function,

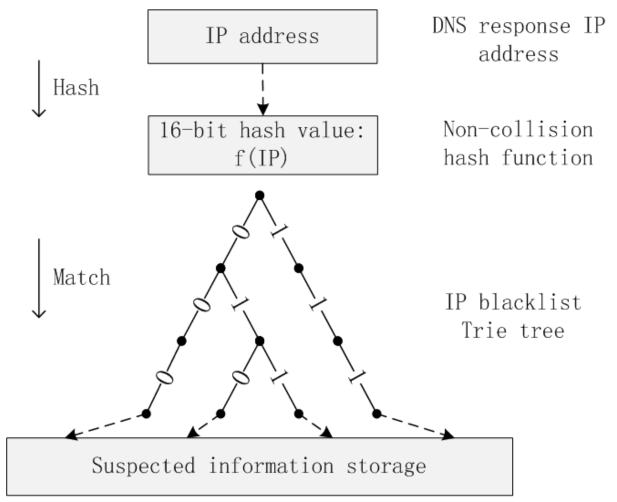

FIGURE 3. Process of multi-bit Trie-tree IP matching algorithm

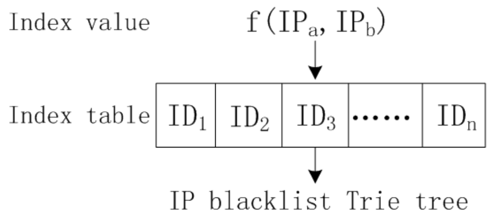

FIGURE 4. Structure of IP blacklist Trie tree

IP_a is the first 16 bits of IP address, IP_b is the last 16 bits of IP address, and ID is the 16-bit index value.

In this algorithm, the hash function includes three parts: block, shift and XOR. The formula is the following:

$$ID = f(IP_a, IP_b) = (IP_a \gg 3) \oplus (IP_b \ll 3) \tag{1}$$

5. Performance Test and Evaluation.

This section presents the performance test and analysis about DNSAM, mainly including three parts: the test of response domain name blacklist, the test of response IP blacklist, and the test of domain name redirecting. The experiments use the mirror data of a switch in the campus network as the test data, and the background traffic is about 800Mbps; besides, the range of DNS data traffic is from 10Mbps to 80Mbps. The configuration of DNSAM is as follows: the operating system is Fedora 8 whose kernel version is 2.6.23, the CPU frequency is 2.4GHz, and the physical memory is 2GB.

5.1. Test of response domain name blacklist.

In order to achieve the successful percentage of response domain name monitoring, we carry out the test of response domain name blacklist for DNSAM under different DNS data traffic (10-80Mbps). Figure 5 illustrates the successful percentage of response domain name monitoring when the number of entries in response domain name blacklist is 10, 100 and 500 respectively, and the string length of each entry is from 5 to 10 characters. From the figure we can see that with the increase of the entries in response domain name blacklist and DNS data traffic, the successful percentage is decreasing accordingly, but always maintains relatively high values. At the same time, we can see that the entry in the response domain name blacklist is an important affecting factor for response domain name monitoring of DNSAM, and this is why we introduce the improved multi-pattern MWM domain name matching algorithm to improve the implementation efficiency.

5.2. Test of response IP blacklist.

About the same as the test of response domain name blacklist, in order to achieve the successful percentage of response IP monitoring, we carry out the test of response IP blacklist for DNSAM under different DNS data traffic (10-80Mbps). Figure 6 illustrates the successful percentage of response IP monitoring when

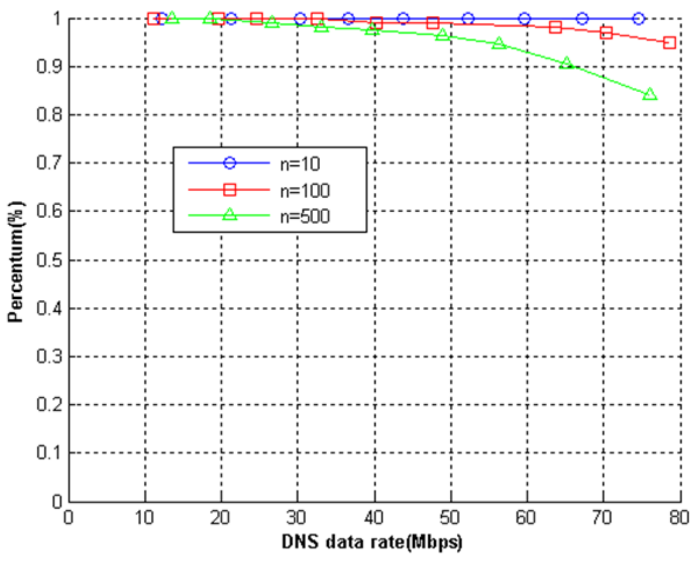

FIGURE 5. Successful percentage of response domain name monitoring

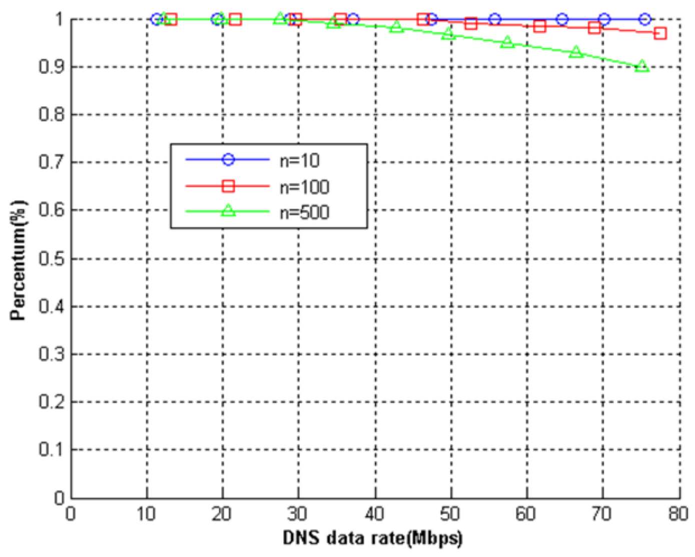

Figure 6. Successful percentage of response IP monitoring

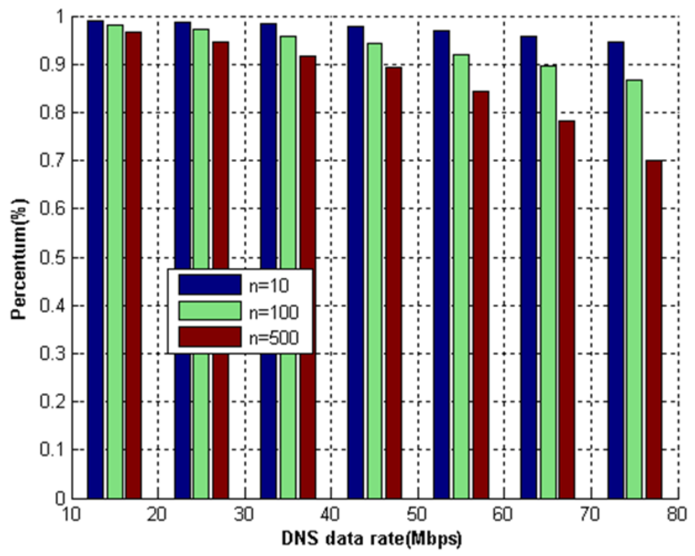

Figure 7. Successful percentage of domain name redirecting

the number of entries in response domain name blacklist is 10, 100 and 500 respectively. From the figure we can see with the increase of the entries in response IP blacklist and DNS data traffic, the successful percentage is decreasing accordingly, but always maintains relatively high values. For example, when the DNS data traffic is about 70-80Mbps and the number of entries is 500, the successful percentage is more than 90%. So DNSAM has a satisfying performance by introducing the multi-bit Trie-tree IP matching algorithm.

5.3. **Test of domain name redirecting.** In this section, we implement the experiments of domain name redirecting to show the successful percentage to redirect the domain name under different DNS data traffic (10-80Mbps). We use a terminal as the host who accesses the vicious Web servers, and its DNS query packets would be mixed in the background traffic. When the number of entries in domain name redirecting blacklist is 10, 100 and 500 respectively, we redirect the DNS query of this terminal and observe the successful percentage. As shown in Figure 7, when the number of entries in domain name redirecting blacklist is 10, with the increase of DNS data traffic the successful percentage to redirect the domain name maintains more than 95%. While the number of entries in domain name redirecting blacklist is 100, the successful percentage to redirect the domain name exceeds 85%. Once the number of entries in domain name redirecting blacklist is 500, the successful percentage to redirect the domain name reaches to 70%. In many practical applications, the illegal DNS query packets hold a very small part of all the DNS query packets; therefore, over 70% of successful percentage can supply a normal demand.

6. Conclusion and Future Work.

This paper has proposed a DNS data real-time analysis and monitoring system – DNSAM. DNSAM uniquely introduces the improved multi-pattern MWM (Modified Wu_Manber) domain name matching algorithm and the non-collision hashing multi-bit Trie-tree IP matching algorithm, and quickly matches the critical information in DNS query and response packets with the DNS security policy. As a third-party network monitoring system, it mainly includes two functions: one is the DNS suspected information storage, and the other is the suspected domain name redirecting. Besides, in order to sufficiently utilize the CPU resources, this system introduces the multi-threading mechanism based on CPU efficiency balancing. At last, we carry out the experiments to illustrate the performance of this system. In the future work we will further research DNSAM and improve the matching algorithm, and its ultimate goal is to meet the needs of heavy DNS data traffic.

Acknowledgments. This work was supported by the Fundamental Research Funds for Central Universities under grant No. 2015JBM008.

REFERENCES

[1] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell and D. Song, Towards a formal foundation of Web security, *The 23rd IEEE Computer Security Foundations Symposium*, Edinburgh, UK, pp.290-304, 2010.

[2] A. D. Rubin and D. E. Geer, A survey of Web security, *Computer*, vol.31, no.9, pp.34-41, 1998.

[3] S. Yu, Y. Tian, S. Guo et al., Can we beat DDoS attacks in clouds?, *IEEE Trans. Parallel & Distributed Systems*, vol.25, no.9, pp.2245-2254, 2014.

[4] *Snort*, http://www.snort.org.

[5] Y.-H. Choi, M.-Y. Jung and S.-W. Seo, L+1-MWM: A fast pattern matching algorithm for high-speed packet filtering, *The 27th Conference on Computer Communications*, Phoenix, USA, pp.2288-2296, 2008.

[6] F. Shang, Y. Pan, X. Pan and B. Bi, Research on a stochastic distribution multibit Trie tree IP classification algorithm, *Journal on Communications*, vol.29, no.7, pp.109-117, 2008.

[7] F. Shang, H. Tang and Y. Pan, Study on an absolute aon-collision hash IP classification algorithm, *Journal on Communications*, vol.26, no.2, pp.87-99, 2005.

[8] R. Arends, Protocol modifications for the DNS security extensions, *IETF RFC 4035*, https://datatracker.ietf.org/doc/rfc4035/, 2005.

[9] J. Kristoff, *An Automated Incident: The Measurement Factory: Dnstop Tool [CP/OL]*, http://dns.measurement-factory.com/tools/dnstop/.

[10] W. Duane, *The Measurement Factory: DSC-DNS Statistics Collector [CP/OL]*, http://dns.measurement-factory.com/tools/dsc/.

[11] W. Z. Khan, M. K. Khan, F. T. B. Muhaya et al., A comprehensive study of email spam botnet detection, *IEEE Communications Surveys & Tutorials*, vol.17, no.4, p.1, 2015.

[12] Y. Pan and X. Ding, Anomaly based Web phishing page detection, *Proc. of the 22nd Annual Computer Security Applications Conference*, Washington DC, USA, pp.381-393, 2006.

[13] R. Kozik and M. Choras, Adapting an ensemble of one-class classifiers for a web-layer anomaly detection system, *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp.724-729, 2015.