Research of deep packet inspection technology in industry control systems based on d-Left counting bloom filter

Kehe Wu, Yi Li*, Long Chen, Fei Chen

Department of Control and Computer Engineering, North China Electric Power University, 102206 Beijing, China

Received 1 May 2014, www.cmnt.lv

Abstract

Aims at the problem of industry control system logic being tampered by malicious programs, this paper proposed a deep packet inspection model for industry control systems. It adopts d-Left Counting Bloom Filter to implement string matching of characteristic database which has a higher matching efficiency and lower misjudgement rate compared to other algorithms. This model can monitor malicious behaviours in industry control systems effectively and can basically meet the demand of security and reliability in current industry control systems.

Keywords: industry control system, deep packet inspection, d-left counting bloom filter, information security

1 Introduction

With rapid development of computer and network technology, industry control systems have been widely adopted in areas such as electric power, water power, petrochemical industry, medicine, food, automobile and aerospace, and they play a vital role in key infrastructure of a country. Thus how to maintain information security of industry control systems is critical to a country's strategic security [1]. In early days industry control system networks are physically separated to external networks, while in recent years, information technology and the demand of enterprise administration improves rapidly, which make industry control system network and external network integrate smoothly. Especially the adoption of Ethernet and TCP/IP in industry control networks enable these systems communicate with external systems directly, even connect to Internet. While this measure pushes industry production forward extremely, it brings many security problems at the same time like Trojans, viruses and information disclosure and control instruction tampering caused by network attacks [2, 3]. Unfortunately, these problems are not paid enough attention until discovery of Stuxnet in 2010 [4, 5]. After broke into the control system of uranium enrichment factory and nuclear power plant in Iran, Stuxnet tampered the control logic of programmable logic controllers lie in centrifuge control system to make the spin frequency of centrifuge change abnormally, which makes the centrifuge break down ultimately [6-8]. In order to prevent industry control system components from being destroyed by malicious programs, we need to monitor network traffic of industry control system to identify and block malicious behaviours [9, 10].

Aims at the problems discussed above, this paper proposed a deep packet inspection model for industry control systems based on d-Left Counting Bloom Filter. It implements inspection of important field or value in data packets through more detailed rule configuration against specific industry control system protocols. This model can monitor malicious behaviours in industry control systems effectively and improves security and reliability in industry control systems.

2 Deep packet inspection technology

2.1 FUNDAMENTAL OF DEEP PACKET INSPECTION

Deep packet inspection (DPI) is a kind of message inspection technology contrary to traditional packet inspection technology [11]. The so-called "depth" means the level of packets being analysed is deeper than the traditional measures, which only analyse content of IP packets below the Application Layer, including the source address, destination address, source port, destination port and protocol type. However, DPI not only has the function introduced on the front, but also increases the Application Layer analysis which can recognize and content of various applications [12]. Although the industry has not formed a relatively clear definition of DPI technology, it is generally believed that, a device with DPI enabled in addition to the analysis with ordinary message inspection ability (i.e. the message header information, including the source IP address, destination IP address, source port, destination port and protocol type), also can be combined with monitoring between factors such as the payload and the message to implement "deep" recognition [13]. The

* ***Corresponding author*** e-mail: liyi174748@163.com

fundamental of deep packet inspection is shown in Figure 1.

In this figure, load balancer is responsible for processing data flow through it, and the position it lies is also the position a firewall lies usually. We take data flow as a group of TCP/IP packets, each packet is independent. In order to ensure data packets arrive in correct sequence, the packets should be reordered before data flow reorganization. Then the load balancer reorganize the data, analyse its protocol in depth, and distribute them to different destination such as Web server, database system, security system, application server etc.. With specific meaning of context, data can be analysed deeply, and can detect attacks from the network layer to the application layer, thus to provide comprehensive protection to computer systems.

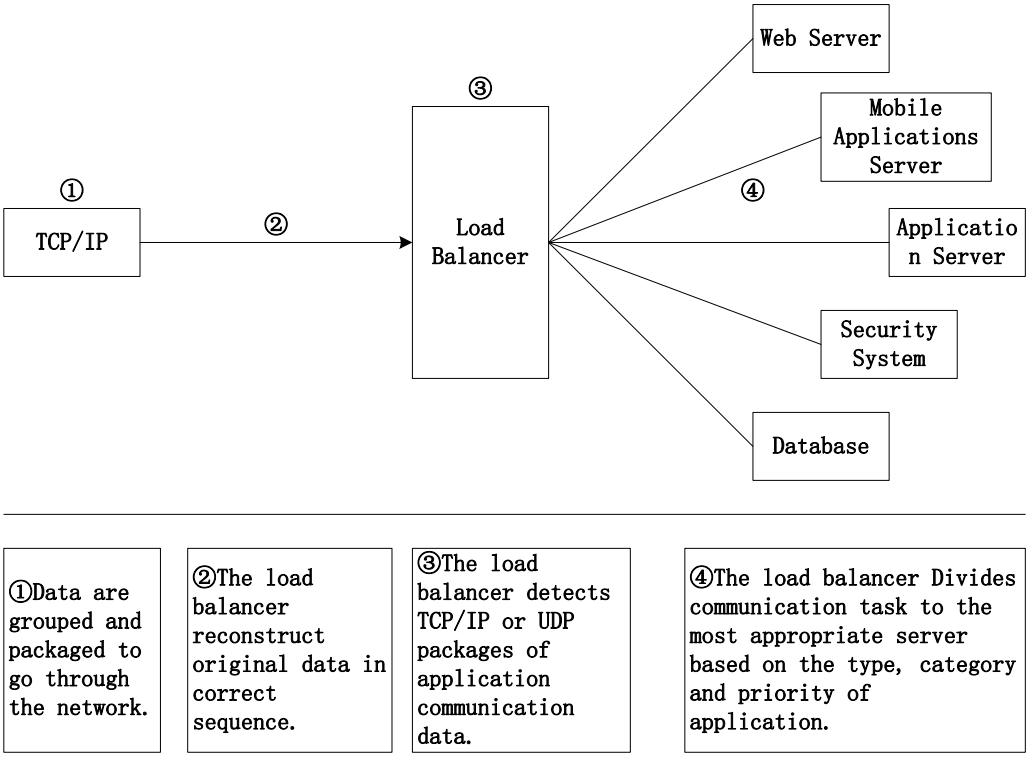

FIGURE 1 Fundamental of deep packet inspection

2.2 ADVANTAGES COMPARED WITH TRADITIONAL PACKET INSPECTION TECHNOLOGY

Although adopts packet filtering to implement packet inspection, but deep packet inspection is different from traditional packet inspection technology, its main advantage is illustrated in the following three aspects.

1) Recognition of protocols in deeper layers.

The traditional packet inspection technology makes the decision of whether forward or block a packet by analyzing its header, for example, it analyses source IP address, source port, destination IP address, destination port, and the protocol used in the IP packet header. If it meets the rules set before, forward it, otherwise it will be dropped. Whereas the deep packet inspection technology is different, it not only can analyse packets on the network layer, but also can analyse packets based on application layer protocols, thus can clearly know what a data bit means in data packets. It is also because of this, a firewall with deep packet inspection can achieve many functions which traditional firewall cannot achieve.

2) Lower Missing Rate

The traditional packet inspection technology analyses and filters separate packet, this approach focus on only part of the overall which makes it easier to miss some attacks, even with state detection. While firewall with deep packet inspection is a real application level firewall, it reorganises packets and can truly realize deep packet content inspection based on state, so it can provide full and complete access control and protection from data link layer to application layer and can reduce missing rate.

3) Stronger Protection Ability

The so-called application layer inspection in some firewalls based on traditional detecting packet inspection just use proxy technology such as URL filtering or other functions in fact, and cannot reconstruct separate packets. For example, if a data flow contains URL bytes is separated into packets, it will not be reverted. That is to say, some malicious programs may bypass this kind of firewalls if skilfully designed. Deep packet inspection technology on the other hand, performs "data reorganization" and "intrusion detection" on the basis of packet filtering, which will be effective to protect computer system from being attacked by application layer attacks [14].

3 Analysis of d-Left counting bloom filter

3.1 PROPOSAL OF D-LEFT COUNTING BLOOM FILTER

Bloom Filter is an algorithm Burton H. Bloom proposed in 1970 to solve the problem of whether a certain or some elements are in the set [15]. It uses *k* independent hash functions to map each element into *k* bits of a *m* bits vector. For each element *x*, the corresponding bit will be set to 1 if the hash function $h_i(x)$ maps into this bit $1 \le i \le k$. If the bit has been set to 1, then the value of this bit won't be changed even it is mapped for the second time. When we query an element *y*, we use these *k* hash functions to map it into *k* bits of the vector. If the value of each bit is 1, we consider this element is in the set, otherwise is not. Bloom Filter is a space saving, efficient data representation and query structure, it uses digit group to represent a collection concisely, and can judge whether an element belongs to the set with a very high probability. However, as hash functions may have collisions, this may cause a fault called "false positive", i.e. it may judge an element in the set which is not in fact. On the other hand, we cannot delete an element from a Bloom Filter, so we cannot perform operations like update and deletion [16, 17].

To solve the problem that elements cannot be deleted in Bloom Filter, Fan L et al. proposed an improved algorithm known as the Counting Bloom Filter (CBF) [18]. It expands the bit vector into a set of counters, and the value of corresponding counter will increase by 1 when it is mapped by a hash function. Conversely the value will decrease by 1 when an element is deleted.

As we know, the counter may overflow and in turn may cause a fault called "false negative", that is to say, it may misjudge an element not in the set. To solve this problem, Bonomi et al. proposed the d-Left Counting Bloom Filter (dLCBF) algorithm [19]. The constructing process of d-Left Counting Bloom Filter is as follows: first use *d* hash tables, each table have many buckets, each bucket can accommodate a number of (fixed usually) cells, each cell has a fixed number of bits used to hold a fingerprint and a counter to deal with collision. Collision should be dealt with in each table. If there is collision occurs in a hash table, i.e. the same fingerprint is stored in the same location, then just increase the value of the counter by 1. When an element is inserted into the set, we first calculate its hash value and divide it into $d+1$ segments, in which d of them are taken as store address corresponding to these hash tables, and the other one is the fingerprint to identify the element. Thus, there are *d* storage locations and a fingerprint. Then we look up these hash tables and choose the leftmost of the lightest load buckets to insert. If the selected position had been stored the same fingerprint, add 1 to the counter of the cell. Conversely in deletion on an element, we get its hash value and find the corresponding counter in the hash table according to the fingerprint and decrease the value of counter by 1. Figure 2 shows the principle of d-Left Counting Bloom Filter algorithm.

Through this algorithm solves the problem of counter overflow may occur in Counting Bloom Filter, it has a flaw may occur in deletion. As we may insert two elements with the same fingerprint and the same bucket index for one of the hash tables, if they are inserted into buckets of different hash tables, when we want to delete one of them, we cannot distinguish which one should be deleted.

To this end, Bonomi et al. introduced a group of random permutations which number is equal to the number of hash tables. After the bucket indexes are divided, we use these permutations to map each bucket index into another value. If we use $\mathrm{H}(x) = f(x) = (b, r)$ in which *b* means bucket index, represent storage location; *r* means remainder, represent the fingerprint to store. Then the permutations can be represented like this:

$$\mathrm{P}_1(f_x) = (b_1, r_1), \mathrm{P}_2(f_x) = (b_2, r_2), \cdots, \mathrm{P}_\mathrm{d}(f_x) = (b_d, r_d)$$

In which $\mathrm{P}_1(f_x)$ means storage position and fingerprint of element *x* for the corresponding hash table. Because permutation means one element will be mapped into another different one, the hash value after the permutation will be different. Thus we can avoid elements from being inserted into the same bucket of a hash table. In process of algorithm realization, linear function can be adopted to realize random permutation. If the range of hash value is 2^q, the random permutation can be written as:

$$\mathrm{P}_\mathrm{i}(H(x)) = aH(x) \bmod 2^q,$$

in which a is a random odd number in range 2^q.

After that, before insertion, we should look up the selected bucket indexes to find whether they have the same fingerprint, if any increase the counter of corresponding cell by 1. Because of the storage locations of different elements are not coincident except there are collisions appeared. Once collision detection is done before insertion, delete operation will not find two identical fingerprint in different locations. This would solve the flaw when deleting elements.

3.2 COMPARISON BETWEEN DLCBF AND CBF

Now we make a comparison between d-Left Counting Bloom Filter and Counting Bloom Filter. Suppose the set contains *m* elements, construct a d-Left Counting Bloom Filter like this:

1) the d-Left hash table contains 4 subtables;

2) suppose average load of each bucket is 6 elements ultimately, thus making each subtable contains *m*/24 buckets;

3) each suitable contains 8 cells, that will be able to guarantee the counter not overflow with high probability;

4) each counter of cell contains 2 bits, so it can hold 4 identical fingerprints. The value of counter should be set to 0 at the first time.

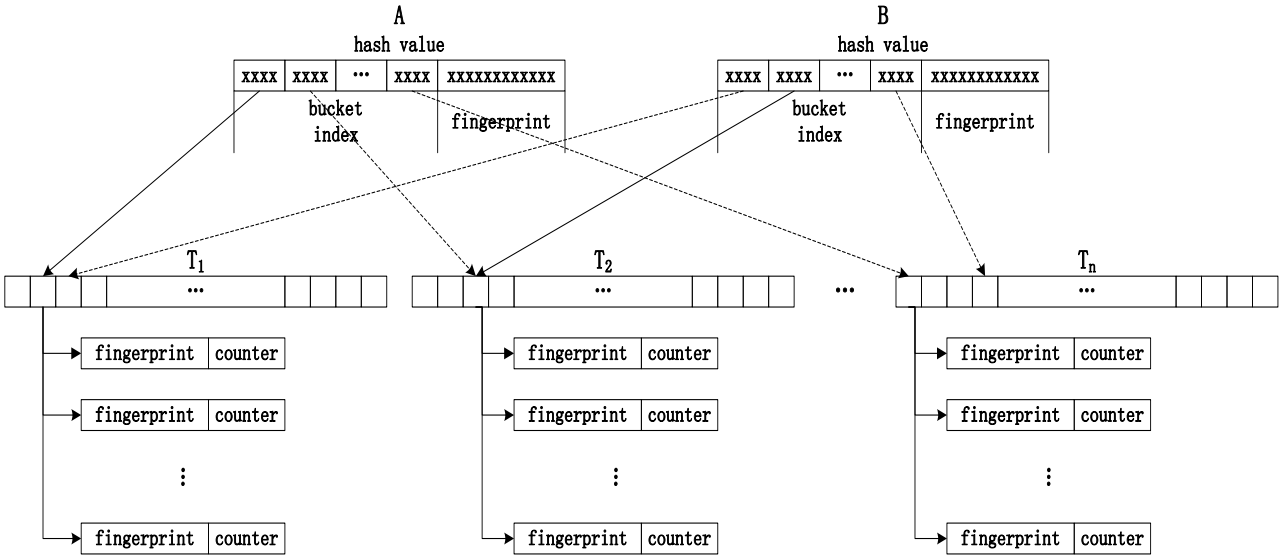

FIGURE 2 Principle of d-Left Counting Bloom Filter

Firstly we compare false positive rate of the two algorithms. Assuming there is a set contains *m* elements. For Counting Bloom Filter, it is constructed by cm counters, in which *c* represents ratio parameter between the number of counters *n* and the number of elements *m*, each counter occupies 4 bits. When the number of hash functions *k* in this algorithm adopts the optimal value $c\cdot\ln 2$, its false positive rate is $\left(2^{-\ln 2}\right)^c$, and it occupies $4cm$ bits in total.

For d-Left Counting Bloom Filter, we use *r* bits to represent the fingerprint. If there is false positive occurs, i.e. there is collision between an element in query and an in the set, the two elements must have the same storage location (the same subtable, same bucket and same cell) and they must have the same fingerprint after hash calculation and permutation. Hence its false positive rate is $\binom{4}{1}\cdot\binom{6}{1}\cdot\left(\frac{1}{2}\right)^r = 24\cdot 2^{-r}$. As each counter have 2 bits, thus each cell occupies $(r+2)$ bits. And because each subtable has *m*/24 buckets, each bucket contains 8 cells, this algorithm needs

$4\cdot\frac{m}{24}\cdot 8\cdot(r+2) = 4m(r+2)/3$ bits in all.

From $4cm = 4m(r+2)/3$ we can acquire when the parameter $c=(r+2)/3$, the bits these two algorithm occupied is equal. Through calculation we can see that when $r\le 6$, Counting Bloom Filter has a lower false positive rate; whereas when $r\ge 7$, d-Left Counting Bloom Filter is lower and the bits of fingerprint occupied grows, the gap of rate between them is bigger and bigger. Figure 3 illustrates the comparison situation.

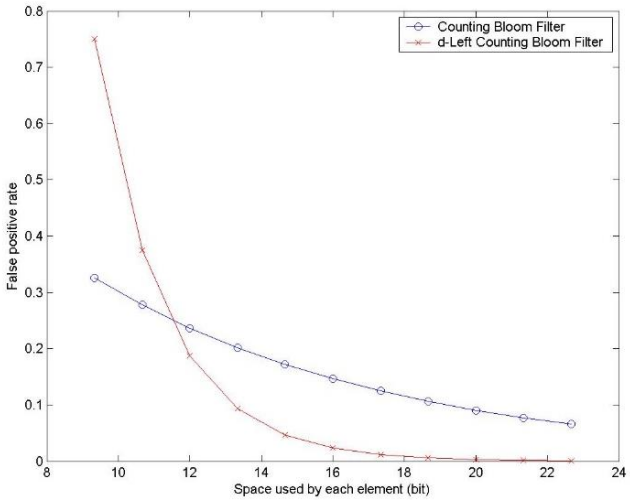

FIGURE 3 Comparison between Counting Bloom Filter and d-Left Counting Bloom Filter

Secondly we compare space occupancy situation of the two algorithms when they have the same false positive rate. Assuming Counting Bloom Filter uses 6 dependent hash functions, each counter occupies 4 bits. According to Bloom Filter optimal number of hash functions $k=c\cdot\ln 2$, if we want the false positive rate lower than ε, the number

of counters n should be at least $k/\ln 2 = 8.66$ times more than the number of elements *m* in set. That is to say, each counter need to adopt 9 counters to guarantee the false positive rate low enough, at this time the Counting Bloom Filter occupies 36*m* bits in total, its false positive rate is 0.01325. For d-Left Counting Bloom Filter, if each fingerprint is represented by 11 bits, it occupies $4m(11+2)/3 = 52m/3$ bits and the false positive rate is 0.01172. The space occupancy ratio of them is $52m/3 \div 36m = 0.48$. In other words, d-Left Counting Bloom Filter achieves a lower false positive rate using just less than half of the space occupied by Counting Bloom Filter.

4 Deep packet inspection model in industry control systems based on d-Left Counting Bloom Filter

This deep packet inspection model is divided into two parts: foreground modules and background modules. The background modules are responsible for recognizing and processing data packets, and recording the condition of handling. Background modules are made up of the characteristic library, traffic recognition module, database module, traffic management module, and log module. The Foreground is responsible for issuing user strategy, and displaying log information [20]. The whole architecture of the model and interactions between each module are shown in Figure 4.

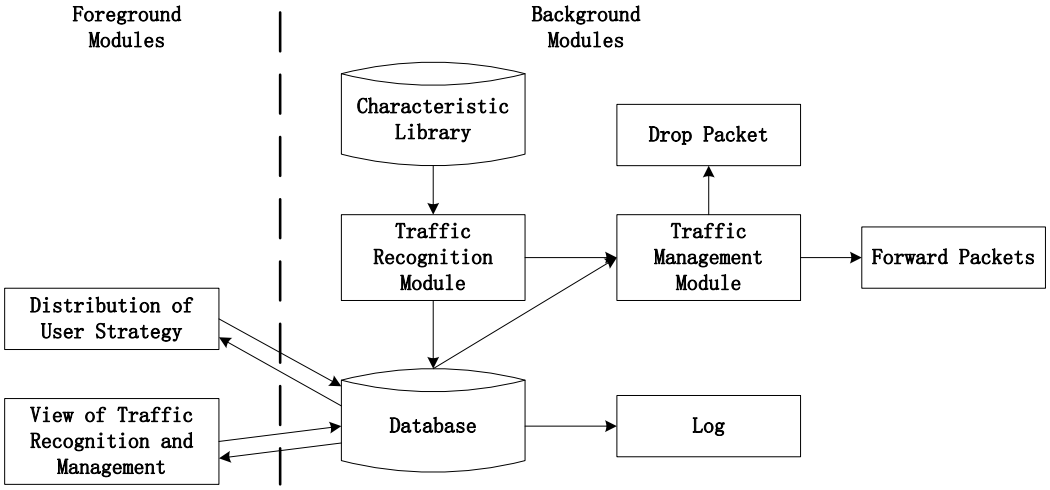

FIGURE 4 Architecture of Deep Packet Inspection Model

The main process of this model is as follows. The packets captured by firewall will be sent to the traffic recognition module then the traffic recognition module loads the access control rules of the characteristics library and perform access control detection to these packets. If they do not meet the requirements of rules the packets will be discarded directly and the recognition results will be stored into database at the same time. Otherwise the packets will be transmitted to the traffic control module, in which recognition strategy is taken out from database module to determine which measure to take on the next step according to the strategy.

4.1 FOREGROUND MODULES

Foreground modules provides display function of the system, mainly including user strategy distribution and viewing traffic recognition and management.

With user strategy distribution module, users can decide IP addresses, protocols, feature strings in packet content and type of control (forward or block). The strategy information will be stored in database after that.

With traffic recognition and management viewing module, users can query traffic recognition results in real time or sometime before. The results also contain traffic amount of corresponding protocol, percentage of total traffic and the type the traffic belongs to etc.

4.2 TRAFFIC RECOGNITION MODULE

In recognition of data traffic, we need to choose different recognition technology according to the characteristic of traffic. To content of some protocols, it can be identified by matching with rules stored in the characteristic library. But to some protocols, we should take other information into consideration like packet size and packet rate, or take load content for deep analysis, hence we should provide specific recognition method for these protocols. According to analysis of industry control system protocols like Modbus and Profibus and some commonly known application protocols, we summarizes the following protocol recognition process, as Figure 5 illustrates [21, 22].

Firstly, we distribute the data packets and check whether they meet the entrance condition, which usually contains common access control parameters like packet size, IP address and port. Then if they meet the entrance condition, we will check whether these packets use known protocols. If they are, the content of packets will be reconstructed and matched with corresponding application protocol rules in characteristic library. If the packets do not

meet entrance condition or they use protocols not known before, the reconstructed content will be matched with specific rules (e.g. feature codes or key words) lie in characteristic library. If the rules are traversed without matching anyone, we consider the recognition results as unknown. Finally, we achieve the recognition results, deposit them into database and transmit them to traffic management module.

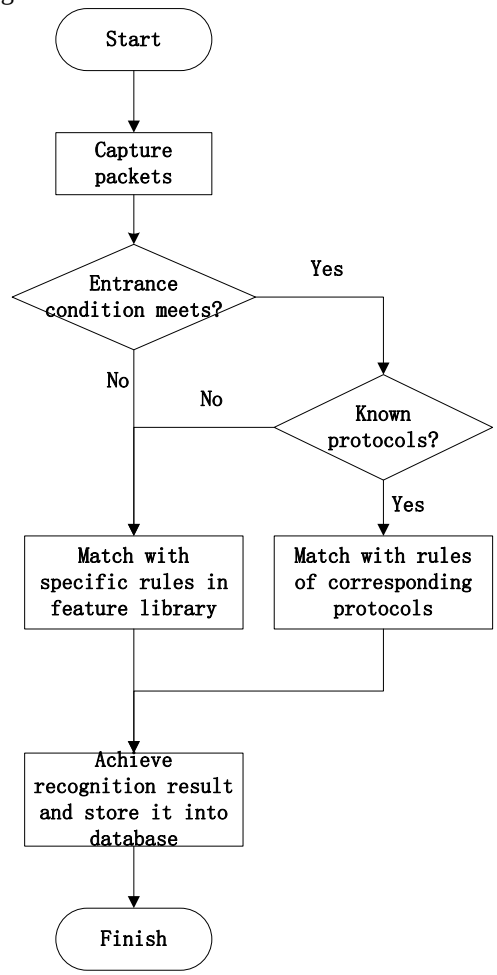

FIGURE 5 Process of Protocol Recognition Module

4.3 TRAFFIC MANAGEMENT MODULE

On system initialization, the traffic recognition module takes user strategies out from characteristic library and put it into a linked list. When there are traffic recognition results achieve, this module determine whether to forward or block the packets according to strategies lies in the linked list. The purpose of doing so is because query in linked list is faster than in the database, but we need to scan the user policy in database regularly, and reload the strategy into the linked list if there are any updates.

4.4 DATABASE

From Figure 4, we can see that the database have interaction with foreground modules, traffic recognition module and traffic management module, in addition it is the data source of the log module. This measure not only avoids much network communication between other modules, but also saves large amount of memory space occupied for data storage.

The database stores various kinds of packet information (e.g. packet size, capture time, header information, load content, recognition result and processing result); the user strategy such as target IP address and protocols. The traffic recognition module deposit recognition results into database for viewing in foreground modules or exporting to log module. Foreground modules put user strategy into database for the use of querying in traffic management module, while foreground modules can view results of recognition and management stored in database, through which avoid direct communication between foreground modules and traffic management module. From this point, the database not only becomes data transmission channel between other modules, but also persistent data.

When there are data packets arrive in traffic management module, if the results of protocol recognition are unknown, it will alert the user to determine whether to forward the packets or block them. Otherwise it will traverse the linked list of strategy, if the strategy corresponding to this protocol order to block this traffic, the packets will be dropped directly; if not, the packets will be forward to user applications. At the same time, the results will be stored to the database; hence can interact between the log module and the foreground module.

4.5 LOG

The function of log module is to record the operation performed by the model and error information, monitor health condition of system, as well as regular export recognition results and condition of blocking from the database for long-term preservation.

5 Conclusions

Focuses on special demand in industry control system application process, through study of deep packet inspection technology and d-Left Counting Bloom Filter in depth, this paper put forward a deep packet inspection model of industry control system based on d-Left Counting Bloom Filter. As d-Left Counting Bloom Filter algorithm has lower space complexity and error rate, it is suitable for implementation in chips; thereby it is convenient for using in high speed integrated circuits integrated into the network equipment, which has a high degree of deployment flexibility with high performance at the same time.

Acknowledgments

The authors wish to acknowledge anonymous reviewers for their constructive comments and suggestions on improving the presentation of this paper. The work of this

topic is supported in part by Department of Control and Computer Engineering, North China Electric Power University. We also get lots of help from Center of Electric Information Technology in Beijing.

References

[1] Berman D, Butts J 2012 Towards characterization of cyber attacks on industrial control systems: Emulating field devices using gumstix technology. In Resilient Control Systems (ISRCS) *5th International Symposium IEEE* 63-8
[2] Shin S 2012 Activities for Control System Security in Japan *In SICE Annual Conference (SICE) IEEE* 667-9
[3] Karnouskos S 2011 Stuxnet worm impact on industrial cyber-physical system security *In IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society IEEE* 4490-4
[4] Cheminod M, Durante L, Valenzano A 2012 System configuration check against security policies in industrial networks. In Industrial Embedded Systems (SIES) *2012 7th IEEE International Symposium IEEE* 247-65
[5] Denning D E 2012 Stuxnet: what has changed? *Future Internet* **4**(3) 672-87
[6] Falliere N, Murchu L O, Chien E 2011 W32 stuxnet dossier *White paper, Symantec Corp Security Response*
[7] Masood R, Um-e-Ghazia U, Anwar Z 2011 SWAM: Stuxnet Worm Analysis in Metasploit *In Frontiers of Information Technology (FIT) IEEE* 142-7
[8] Byres E, Ginter A, Langill J 2011 How Stuxnet spreads–A study of infection paths in best practice systems *White paper*
[9] Faily S, Flechais I 2011 User-centered information security policy development in a post-stuxnet world. In Availability, Reliability and Security (ARES) *2011 Sixth International Conference IEEE* 716-21
[10] Kriaa S, Bouissou M, Piètre-Cambacédès L 2012 Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments. In Risk and Security of Internet and Systems (CRiSIS) *2012 7th International Conference IEEE* 1-8
[11] Riley M C, Scott B 2009 Deep Packet Inspection: The end of the internet as we know it
[12] Building Intelligent Mobile Data Services using Deep Packet Inspection Napatech 2011
[13] Wang J, Wei J, Lin S, Huang W 2013 Research of Deep Packet Inspection System Based on the MapReduce *Journal of Computational Information Systems* **9**(7) 2587-94
[14] Zhou Y 2012 Hardware acceleration for power efficient deep packet inspection *Dublin City University*
[15] Dharmapurikar S, Krishnamurthy P, Sproull T, Lockwood J 2003 Deep packet inspection using parallel bloom filters. In High Performance Interconnects *Proceedings 11th Symposium IEEE* 44-51
[16] Pal S K, Sardana P 2012 Bloom Filters & Their Applications *International Journal of Computer Applications Technology and Research* **1**(1) 25-9
[17] Zink T, Waldvogel M, Scholl M H 2009 Packet forwarding using improved Bloom filters *Master's thesis, University of Konstanz*
[18] Fan L, Cao P, Almeida J, et al 2000 Summary cache: a scalable wide-area web cache sharing protocol *IEEE/ACM Transactions on Networking (TON)* **8**(3) 281-93
[19] Bonomi F, Mitzenmacher M, Panigrahy R, Singh S, Varghese G 2006 An improved construction for counting bloom filters. In Algorithms–ESA *Springer Berlin Heidelberg* 684-95
[20] Byres E, Karsch J, Carter J 2005 NISCC good practice guide on firewall deployment for SCADA and process control networks *National Infrastructure Security Co-Ordination Centre*
[21] Protecting Industrial Control Systems - Recommendations for Europe and Member States ENISA 2011
[22] Samson 1999 PROFIBUS-PA Technical Information, Part 4 *Communication, Frankfurt: Samson AG*

Authors

Kehe Wu, born on August 11, 1962, Jiangsu, China

Current position, grades: professor, Department of Control and Computer Engineering, North China Electric Power University.
University studies: North China Electric Power University.
Scientific interest: intelligent software, cloud computing, information security.
Publications: 80 papers.
Experience: director of the Chinese Association for artificial.

Yi Li, born on October 27, 1988, Shandong, China

Current position, grades: Ph.D. student of North China Electric Power University.
University studies: North China Electric Power University.
Scientific interest: information security, cloud computing.
Publications: 4 papers.

Long Chen, born on March 31, 1988, Shaanxi, China

Current position, grades: Ph.D. student of North China Electric Power University.
University studies: North China Electric Power University.
Scientific interest: cloud computing, distributed storage and networking.
Publications: 4 papers.

Fei Chen, born on April 28, 1986, Jiangsu, China

Current position, grades: Ph.D. student of North China Electric Power University.
University studies: North China Electric Power University.
Scientific interest: electric information security and trusted computing.
Publications: 9 papers.