*Proceedings of the Asia-Pacific Advanced Network **2011** v. 32, p. 45-52.*

The Design and Implementation of the Defender Cloud on TWAREN Backbone

Ming-Chang Liang, Hui-Min Tseng, Shin-Ruey Hsieh, Wei-Jie Liau, Jyun-Jie Chen, Te-Lung Liu, Jee-Gong Chang

National Center for High-performance Computing / No. 28, Nan-Ke 3rd Rd., Hsin-Shi Dist., Tainan City, Taiwan, R.O.C. 74147

E-Mails: {liangmc,n00hmt00,hsiehsr,wjl,jjchen,tlliu,n00cjg00}@nchc.narl.org.tw

Tel.: +886-6-5050940#724; Fax: +886-6-5055909
http://dx.doi.org/10.7125/APAN.32.6   *ISSN 2227-3026*

Abstract: Defender Cloud is a cloud based backbone network defending system having full scope over the whole backbone network. Rather than detecting suspicious network activities on a local area network, it collects and integrates the flow data from all connecting members and all entrances of a backbone network. After analyzing by a proposed cloud based distributed processing model, the corresponding defensive reaction can be carried out in a global basis. Thus its protection can cover the whole network, even including member institutions without their own firewall. This paper illustrates the design, verification and future perspective of the Defender Cloud, with an emphasis on the distributed processing of the flow data.

Keywords: cloud computing; distributed processing; information security; backbone defending; TWAREN.

1. Introduction

As the TWAREN [1] network operating center, we have done various researches and developed network management solutions to increase the availability of the backbone network [2, 3]. Meanwhile, we started to realize the unique value of network security solutions in the backbone network perspective.

TWAREN is a national research and education network. Through its points of presence (POPs) and connecting members, TWAREN reaches most places throughout the country. The widespread of TWAREN enables the opportunity to provide network security solutions in a global and cost effective way. Before the network backbone takes part in the security, most schools and connecting institutions have to buy their own firewalls and intrusion detection systems, which are usually expensive, especially for those models capable of handling very high network bandwidth. Even with such equipment, there is no mechanism to integrate them thus all schools still defend alone. Although the Information and Communication Security Technology Center of Taiwan provides incident alert service, it is operated manually and lacks systemic integration.

In contrast, our cloud based backbone joint defense, the Defender Cloud, collects flow data from all POP's equipment and collaborating schools. Once an intrusion or attack is detected in one place, it will be presented and all related flows will be identified and analyzed globally. All other victims around the whole network will be located and notified. In cases of denial of service (DOS) attacks, since the source IP may be faked, only a backbone level defense mechanism like Defender Cloud can trace back to and block it from the real entrance point. Therefore this is the most significant advantage of the Defender Cloud because all in campus firewalls don't have the global view of the flow. If properly authorized, the Defender Cloud can also actively block those suspicious victims to prevent endangering the whole network by those cracker controlled victim computers. Therefore schools with or without network security equipment all benefit from the backbone based defense mechanism. The Defender Cloud alleviate the security pressure of all connecting members, thus it becomes the most efficient way to increase the security of the whole research and education network.

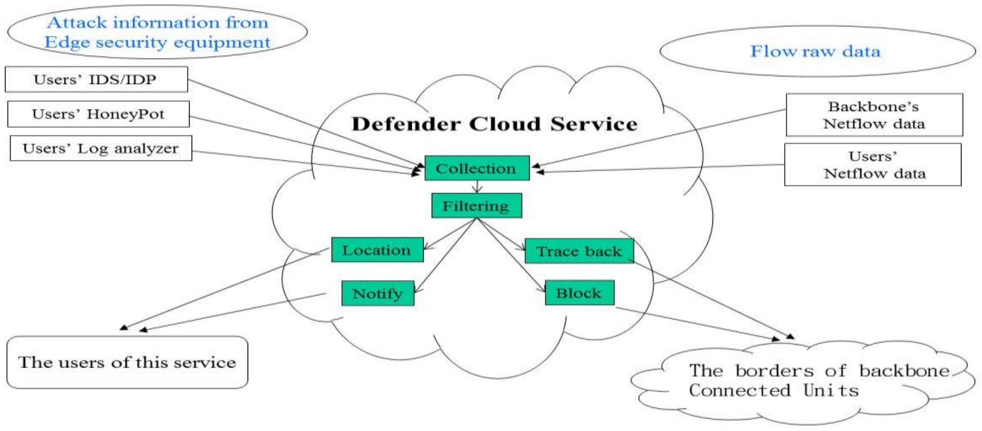


Figure 1. The backbone view of the Defender Cloud.

2. The encountered issues

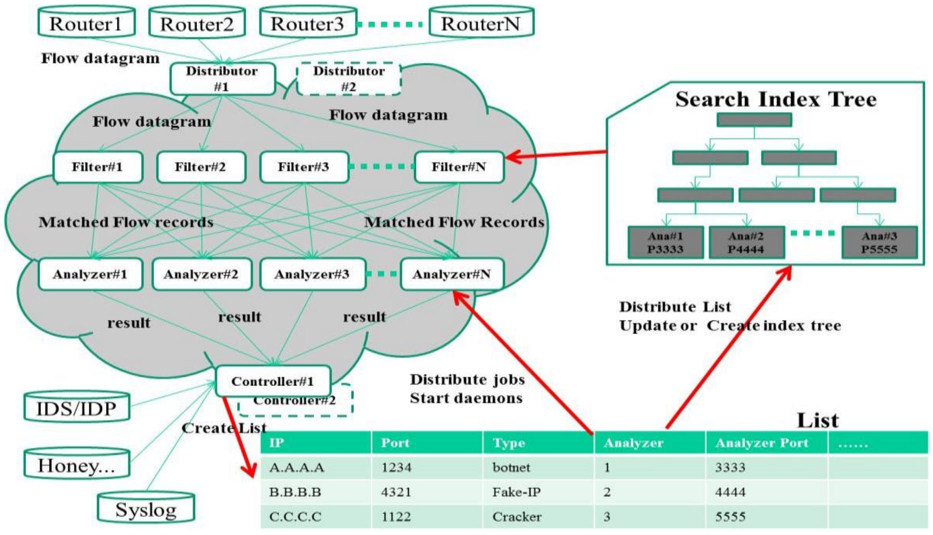
The first issue is the huge volume of flow data. While collecting flow data from all POPs ensures the completeness of information, the data volume is extremely huge. Even the second largest VPN on TWAREN, the research network, could generate 6.22 million flows in 5 minutes on June 30, 2010, which means 20,000 to 30,000 flows per second. It was in the summer vacation; therefore the volume can be significantly larger during semesters. Furthermore, the largest VPN, the TANet, can be still several times larger. The processing of this kind of data volume significantly exceeds the capacity of a single computer, thus a distributed computing model is definitely necessary.

The second issue is the insufficient engineering capacity to deduce the characteristics and traits of malicious activities out from the huge flow data pool and convert them into feasible filter rules. Although the commercial products have already done a good job on detecting malicious activities, using their output as sources needs extensive programming to accommodate the different interfaces and formats of different equipment. Another alternative is to train the 7x24 full time operators of the TWAREN NOC to manually write filter rules out from the network security equipment outputs. Although not fully automatic, the participation of human intelligence yields higher quality of results.

To overcome all these issues, the Defender Cloud must be designed as a distributed system and capable of loading all modules on the fly therefore the analyzing modules can be independently developed and come into effective at any time when loaded.

3. The cloud based architecture design

The architecture of the Defender Cloud is shown in figure 2. Of which the cloud represents the distributed processing platform of the defense system. The system has two inputs. One for flow data, as shown on the upper side of the cloud, and another for target searching filter, as shown on the lower side of the cloud.



IP	Port	Type	Analyzer	Analyzer Port	……
A.A.A.A	1234	botnet	1	3333	
B.B.B.B	4321	Fake-IP	2	4444	
C.C.C.C	1122	Cracker	3	5555	

Figure 2. The Defender Cloud Architecture.

The flow data, composed of flow datagrams from all routers, enter the system through a virtual entrance, the dispatcher, as shown in the figure 2. The dispatcher may consist of multiple load balancers and a DNS to split the incoming flow data to multiple filter machines behind the load balancers. The key role of the dispatcher is to split the load as quickly as possible to avoid becoming a bottleneck in the whole system.

Typically routers send flow data in UDP datagrams. Each UDP datagram contains several flow records. Since routers don't include their IP or any other information capable of identifying themselves in the datagram header of flow version 5, the dispatcher is designed to modify the source IP of the UDP packets as the incoming router IP to indicate its real source.

After receiving the flow datagram, the filter will firstly record the aforementioned router IP from the source IP. Then the srcaddr and dstaddr records of each flow records will be sent into the blacklist comparison tree, respectively. If the srcaddr or dstaddr matches any records in the blacklist, the corresponding flow record will be packaged into an independent UDP datagram and then sent to the designated IP and port (the analyzer) according to the matching blacklist comparison tree. Multiple analyzer programs run simultaneously to analyze the filtered flow record datagram, with each type of analyzer having its own rules and targeting at a certain type of flow record. The results of analyzing will be reported to the controller.

The fundamental idea of the Defender Cloud is to take advantage of the distributed processing model to accommodate the huge volume of the flow records. All the dispatcher, filter and analyzer layers can be constructed by numerous machines to boost the performance. Furthermore, although the analyzing task is usually very complicated, the three layers design ensures that most irrelevant flow data have been filtered by the filter layer, thus only a very small fraction of the flow reach the analyzer, which minimize the possible performance issue.

Table 1. Example of the blacklist.

IP	Port	Type	Analyzer	Analyzer Port	……
A.A.A.A	1234	botnet	1	3333	
B.B.B.B	4321	Fake-IP	2	4444	
C.C.C.C	1122	Cracker	3	5555	

Another key design of the Defender Cloud is to transmit all data with UDP datagram. Without the TCP ACK mechanism, the transmission performance is not affected by the distance and the network delay. Thus the infrastructure of the cloud can then be easily separated into different locations to strengthen the system flexibility without performance loss. In addition, UDP datagram transmission does not need the TCP three way handshaking, which makes the modification to the source IP of the UDP datagram possible. It brings the source router information all the way to the analyzer.

To establish the filter rules, the output of network security equipment from collaborating schools is used and processed by the controller to establish the blacklists. A simplified example of the blacklist is shown in the table 1. The controller categorizes those malicious activities into types like Botnet, DOS fake source IP or cracker, and then assigns each type to a specific analyzer port and respective analyzer program. Even the same category of threat (for example, the Botnet C&C) may be categorized as different types according to their different behaviors..

As the example in table 1, the controller asks the analyzer machine #1 to run the analyzer program #4 on port 3333, to analyze the flow records related to the IP A.A.A.A of Botnet C&C type, to find out the possible infected IPs, and analyzer machine #2 to run analyzer program #12 on port 4444 to process the B.B.B.B related flow records, to back trace the DOS attack to its entrance.

The controller updates the blacklist to the filters such that the filters can build a new blacklist comparison tree accordingly. The structure of the blacklist comparison tree is shown in figure 3. The tree comprises 6 layers of nodes to represent IP and port.

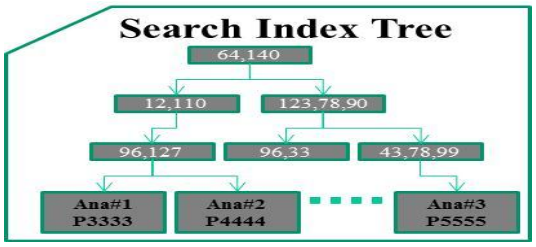


Figure 3. The diagram of the black list comparison index tree.

While filtering, the IP and port of the flow data are compared against the corresponding layers of the tree. This implementation ensures that the time complexity of the blacklist comparison will be constant regardless the size of the blacklist comparison tree. The drawback is the higher demand of the computer memory to hold the complete tree, which is usually not a problem. Once a flow can match to the end node of the tree, it will be repackaged and sent to the analyzer port specified in the end node of the tree.

The analyzer program listening on that specific port is specialized in analyzing that specific type of flow. As controlled by the controller, the analyzer machine can run any type of analyzer program on specific port to serve each type of flow respectively, and then reports the results back to the blacklist assigned port of the controller. Finally the results are visualized on the controller's graphical user interface (GUI) as shown in figure 4.

4. Current implementation

We set up an experimental environment consisting of one dispatcher, two filters, two analyzers and one controller. Since the commercial load balancer is fully competent to be a dispatcher, we used a load balancer instead to focus more on developing the real time filter engine.

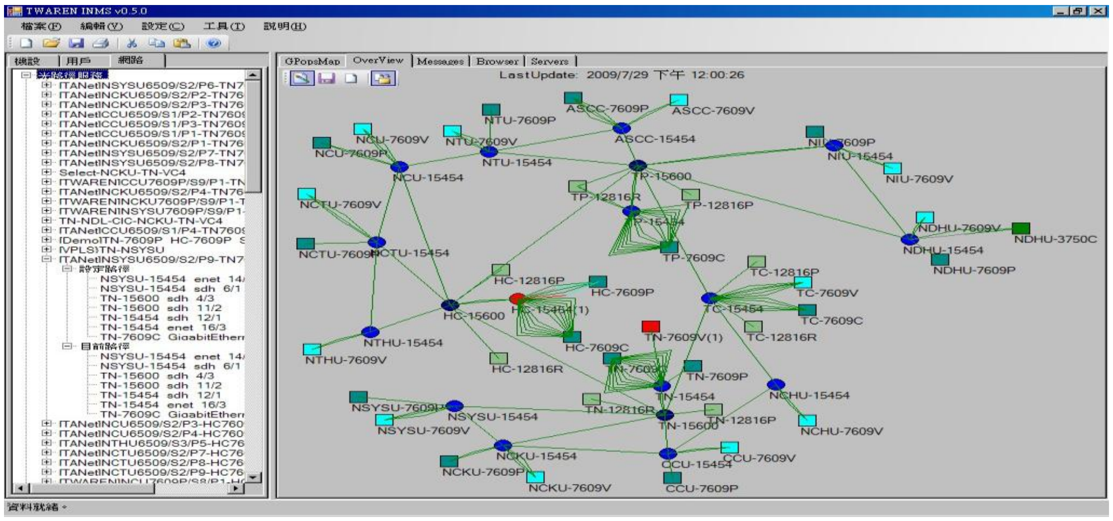


Figure 4. The screenshot of TWAREN network management system.

Since the main goal of the Defender Cloud is to capture the malicious activities as real-time as possible, the filter must continuously process the flow data as soon as possible. If the clearance rate is lower than the flow data incoming rate, the flow data will accumulate and eventually overflow. Therefore the overall processing capacity of the filters must be greater than the total flow data volume. To evaluate the capacity of the experimental system, a series of experiments have been conducted.

First of all, the memory demands of the blacklist comparison trees against the blacklist sizes are tested. Since the memory consumption of the tree is proportional to the blacklist rule numbers, we constructed the trees with randomly generated IP numbers. Because random numbers have very low similarity between each other, it creates the largest possible tree with given blacklist size. From this test we concluded that every 10,000 blacklist rules take roughly 40 Mbytes. So a machine with 1 Giga Bytes memory can do filtering with a blacklist of up to 250,000 rules. This demand should be easily satisfied by the RAM size of typical computers.

The next experiment is to evaluate the performance of the experimental filter system. The experiment was conducted with HP ProLiant DL380 G3 server with Intel Xeon CPU 3.06 GHz and two DIMM DDR 266 MHz (3.8 ns) 512 Mbytes RAM modules (totally 1 Giga Bytes).

Various sizes of randomly generated flow data were filtered against a random number constructed blacklist tree of 5000 rules. The result is shown in figure 5.

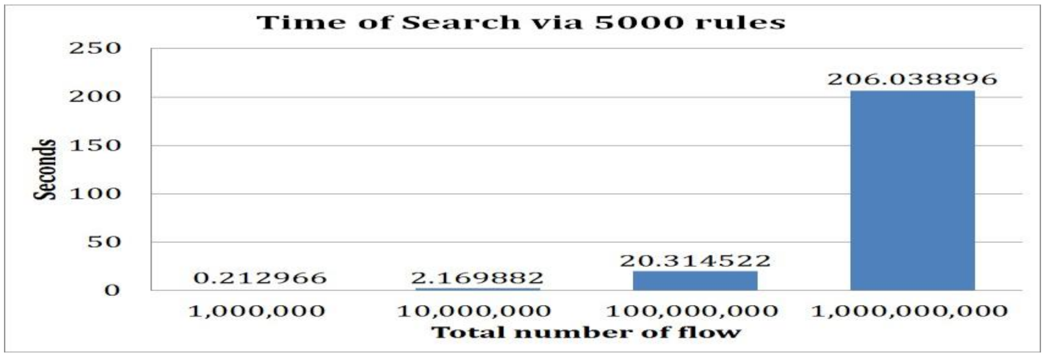


Figure 5. Time consumption of filtering one million to one billion flows.

The results suggested that the filter time consumption increased proportionally to the increase of the flow numbers. When under the same condition, our HP DL380 G3 server is capable of handling 4.77 million flows per second. The flow was randomly generated during runtime in order to approximate the overhead induced by the handling of real flow. Thus we concluded that an ordinary server of the same level will be able to handle several million flows per second.

The last experiment is to evaluate the filter time consumption of 100 million flows over different sizes of blacklist trees. The result is shown in figure 6.

Theoretically the tree size has no significant influence over the comparison performance. The results revealed that the addition of every 10,000 rules increased the processing time by roughly 4 seconds. Since our previous experiment showed that every additional 10,000 rules increase the tree size by 40 Mbytes, the processing time increase may attribute to the additional system loading of memory page operations. In average, the filter spent 170 nanoseconds on each flow. The CPU cycles used on the processing of one flow are not significantly higher than the memory operation time involved. Although the tree size still has some effect on the overall filtering performance, we concluded that the tree size penalty to the performance is negligible before the tree grows to more than one million rules.

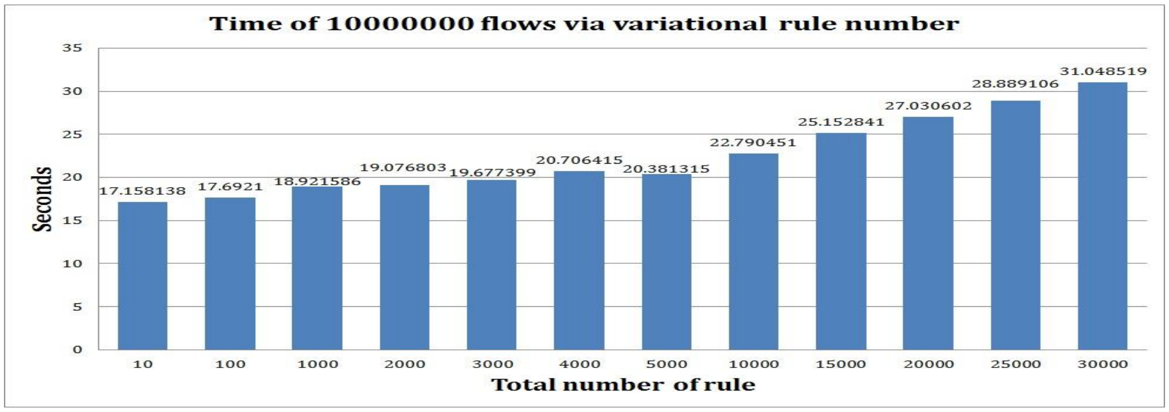


Figure 6. Time consumption of filtering against 10 rules to 30,000 rules.

5. Future perspective

In our current implementation, there are still some missing parts yet to be developed in the future. The first one is the automatic module to convert the inputs from existing network security equipment into filter rules. The possible sources include the Botnet C&C or DOS reports from the IDS/IDP equipment and syslog. The latter will be easier to implement because the format is uniform and well known. However the information retrieval from network security equipment is much harder due to the variation of interfaces and data formats. Solutions will be investigated in the future. The second part is the development of more analyzer. Although it only receives classified inputs from filters, the algorithms to automatically analyze the flow are quite challenging to develop.

The biggest issue restricting the possibility of the Defender Cloud is the limited scope of the flow data because some malicious activities can hardly be detected solely from the flow information.

Due to the distributed and modular design, the analyzers of the Defender Cloud can be written and executed independently. Once conforming to the protocol standard, they can be developed with different programming languages, which provide the best flexibility to the development team. This is the best development model for amateur programmers of the NOC team.

Acknowledgements

We thank the anonymous reviews for their detailed suggestions and comments. This study is supported by the grant from the National Science Council of the Executive Yuan of Taiwan (Project number: NSC99-2218-E492-006).

References

1. TaiWan Advanced Research and Education Network. (http://www.twaren.net/).
2. Ming-Chang Liang; Li-Chi Ku. Design and Implementation of TWAREN Hybrid Network Management System. *APAN2008 Fall Meeting* at Queens Town, New Zealand, 2008.
3. Ming-Chang Liang; Sheng-I Chang; Meng-Chang Lin; Chen-Tsai Chiang. The advanced implementation of network management system with fault detection in TWAREN hybrid network. *APAN2009 Fall Meeting* at Kuala Lumpur, Malaysia, 2009.