Design and Implementation of Proactive Firewall System in Cooperation with DNS and SDN

Tomokazu Otsuka[1], Nariyoshi Yamai[2], Kiyohiko Okayama[3], Yong Jin[4], Hiroya Ikarashi[5] and Naoya Kitagawa[6]

[1] Graduate School of Natural Science and Technology, [3] Center for Information Technology and Management,
Okayama University
3-1-1, Tsushima-naka, Kita, Okayama 700-8530, Japan
[2,6] Institute of Engineering, [5] Graduate School of Engineering, Tokyo University of Agriculture and Technology
2-24-16, Koganei, Tokyo 142-8588, Japan
[4] Global Scientific Information and Computing Center, Tokyo Institute of Technology
2-12-1 O-okayama, Meguroku, Tokyo 152-8550, Japan
E-mail: [1] otsuka.net@s.okayama-u.ac.jp, [2] nyamai@cc.tuat.ac.jp, [3] okayama@okayama-u.ac.jp,
[4] yongj@gsic.titech.ac.jp, [5] hikarashi@net.cs.tuat.ac.jp, [6] nakit@cc.tuat.ac.jp

Abstract: Recently, unauthorized accesses from the external networks to the internal hosts are sharply increasing. Although many firewall appliances are widely utilized as one of the countermeasures, its throughput is not high enough especially when it performs deep packet inspection. In order to solve this problem, we propose a proactive firewall system which consists of two or more firewall appliances with Software Defined Network (SDN) adaptively choosing a proper one for each communication flow based on, for example, whether its peer is trusted or not. To obtain the peer IP address in advance, the system introduces EDNS Client Subnet option of DNS. According to the performance evaluation results on the prototype system, we confirmed that the prototype system could separate flows of trusted hosts from other flows effectively.

Keywords—**Firewall, DNS, SDN, Traffic Engineering**

1. Introduction

Since the attempts of malicious accesses and attacks from the external networks to the internal hosts or networks are sharply increasing today, most organizations introduce some firewall or UTM (Unified Threat Management) appliances (hereafter firewalls for simplicity) as one of the solutions to protect their hosts and networks from those external accesses and attacks.

However, most of these firewalls have to avoid high-load inspection such as Stateful Packet Inspection (SPI) [1] and Deep Packet Inspection (DPI) [2] properly or monitor only suspicious traffic since those inspections can possibly cause decrease of the firewall performance and throughput. Furthermore, the network administrators have to bear a heavy burden to deploy the policies on the firewall system manually based on the layer 3 and 4 information and only pre-defined communications can be controlled by the policy-base firewall system.

To solve these problems, we propose a proactive firewall system which consists of two or more firewall appliances with Software Defined Network (SDN) adaptively choosing a proper one for each communication flow, based on the source and destination information. This system obtains flow information in advance by introducing EDNS Client Subnet option [3] of DNS. By separating normal communication from suspicious communication, the proposed firewall system not only can support high performance for the clear communications but also can decrease the administrative cost.

In the rest portion of this paper, we describe the design of the proposed firewall system in Section 2. In Section 3, we describe implementation and evaluation of the proposed firewall systems. Finally in section 4, we conclude this paper and introduce some future works.

2. Design of the Proactive Firewall System

2. 1 Network topology

As shown in Figure 1, we consider that the target network topology consists of multiple firewalls (FW1, FW2) and a pair of Layer 3 Switches (L3SWs) or Load Balancers (LBs). In this figure, we assume that these FWs perform different inspection. For example, FW1 performs DPI but FW2 does not. L3SWs/LBs forward the packets of the specific flows to the same firewall appropriately, based on layer 3 and layer 4 information such as the source and destination IP addresses as well as port numbers.

Such a network topology is well-known as a typical configuration in most organizations and we also consider it is easy to setup a similar network environment for many other organizations.

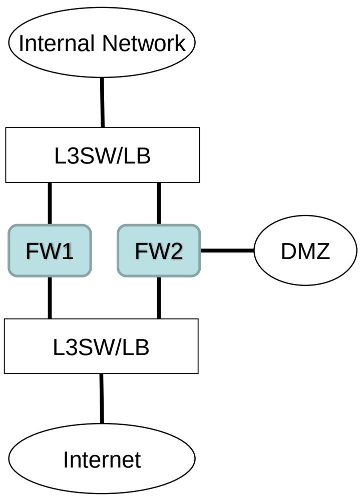

Figure 1 Target network topology.

2. 2 Basic idea of the proposed proactive firewall system

Prior to most TCP/IP communications, domain name resolution with Domain Name System (DNS) occurs. In current DNS protocol, the information of the client which initializes the query is not included in the query message and due to the existence of the DNS cache server it is not guaranteed that the server side DNS server can receive all the client information under the client side DNS cache server. Nonetheless, the characteristic that most clients launch domain name resolution just before the application layer communications is very important. Accordingly, we focus on this characteristic and expand the DNS protocol by adding the function of embedding the client IP address into the query message when performing domain name resolution. Moreover, we also add another function to solve the caching problem and make the server side authoritative DNS server receive all the client IP address. Consequently, the firewall system in the authoritative DNS server side is able to check both the source and destination IP addresses of the communication which is going to happen later and also can dynamically change its policies appropriately.

2. 3 Overview of the proactive firewall system

As mentioned in the previous section, we consider a firewall system consisting of multiple firewalls performing different inspection. To make all packets of a flow go through a specific firewall, we use an SDN controller and two SDN switches instead of L3SWs or LBs.

To choose an appropriate firewall for a flow based on layer 3 and layer 4 information, we introduce a kind of blacklists and/or whitelists internally or externally. For example, an administrator of the firewall system may provide a list of trusted hosts as a whitelist. He/she also may use an external DNS Block List (DNSBL) such as Spamhous XEN [3]. We can also use other services such as GeoIP services [4], which replies the country associated with a given IP address, as implicit blacklists or whitelists based on the peer’s country.

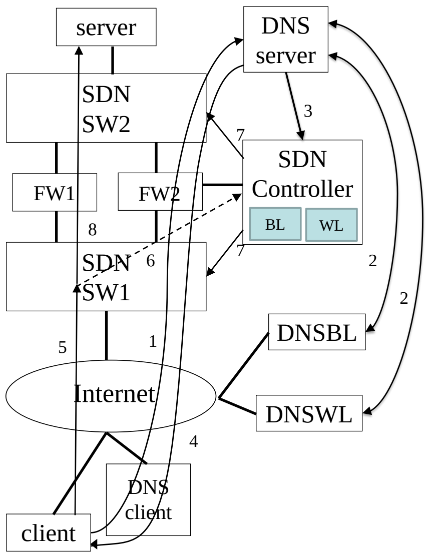

Figure 2 A proactive firewall system configuration.

A typical configuration of the proposed proactive firewall system is shown in Figure 2. In this figure, we assume that FW1 is configured for conventional check for the communications between an internal host and a trusted host registered in the whitelist while FW2 is set for high-load inspection for other communications. Besides, at the initial stage of the entire system, all communications are configured routing to the FW2 by the SDN Controller.

Under this circumstance, the procedures of an example that a client in the Internet accesses an inside server in the organization network are described in the following and the step numbers are corresponding to the numbers in the figure. Here, we mainly describe the name resolution procedures and firewall policies while the detail of application protocol will be omitted.

1. A client in the Internet requests the IP address (A record) of the server to its client-side DNS cache server (DNSclient). Then, the DNSclient sends to the server-side authoritative DNS server (DNSserver) the queries incuding the IP address of the client.
2. When the DNSserver receives a query, it checks if the query message includes the client IP address. If it does include the client IP address, the DNSserver check if the DNSBL and DNSWL have it registered, otherwise the procedure goes to the step 4.
3. The DNSserver registers the client IP address into the temporary whitelist (WL) or blacklist (BL) in the SDN Controller according to whether the DNSBL and DNSWL have the client IP address registered.
4. The DNSserver replies the server information such as the A record to the DNSclient. Note that the DNSserver may reply an IP address of the honeypot or a black hole IP address (a special IP address for dropping the incoming packets) instead of the real server IP address if the client IP address is registered in DNSBL. After that, the DNSclient replies the messages received from the DNS server to the client.
5. The client starts to communicate with the server. The first packet arrives at SDN Switch 1 (SDN SW1).
6. SDN SW1 asks the SDN Controller how to process the incoming packet.
7. The SDN Controller determines which firewall the incoming packet should go through or to discard the incoming packet silently, consulting with WL and BL. Then the SDN Controller registers the flow entry of the incoming packet into both SDN Switches.
8. SDN SW1 processes the incoming packet according to the flow entry. Afterward, all the packets of the same flow are processed in the same manner.

2. 4 Notification of the client IP address [5]

In order to notify the client IP address to the server side authoritative DNS server, we use Client Subnet option [3] of DNS extension. In this method, the client side DNS cache server embeds the client subnet option (the subnet address and subnet mask of the client) into the query message before sending it out. Basically, this method considers the network address notification but if we set the subnet mask with 32-bit it is also possible to notify the entire IP address to the server side authoritative DNS server.

In the proposed system, the client side DNS cache server needs to query the server side authoritative DNS server everytime when the client requests the name resolution for contents even if they are resolved before. However, if the caching function works in the DNS cache server, then the proposed system may not work as we expected since the same name resolution request from a different client may hit the cached record in the client side DNS cache server without notifying the different client IP address to the server side authoritative DNS server.

In order to solve this problem, we consider to disable the partial caching function in the query side DNS cashe server. That is, when the query side DNS cache server receives the same request of name resolution from a different client then it ignores the old information including A , AAAA records and queries to the server side authoritative DNS server directly again. The detail of this function is discussed in [5].

3. Implementation and Evaluation of the Prototype Systems

3. 1 Implementation of the prototype systems

We have implemented two prototype systems, one for functionality evaluation at Okayama University (OU hereafter) and the other for performance evaluation at Tokyo University of Agriculture and Technology (TUAT hereafter). The layouts of two prototype systems, including clients and DNS cache server, are almost the same, as shown in Figure 3. Since we did not have any real firewall appliances, we use two servers, namely Server 1 and Server 2 instead of FW1 and FW2 of Figure 1, respectively. The IP addresses shown in Figure 3 are those for functionality evaluations in a local network environment.

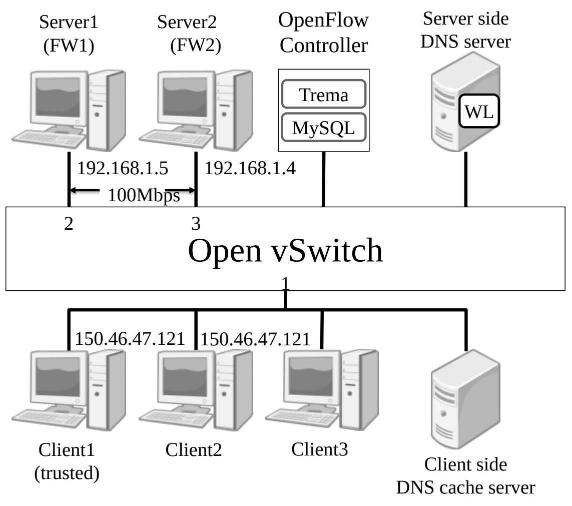

Figure 3 The layout of the prototype system (OU).

The procedure of the prototype system is almost the same as mentioned in Section 2.3, except for having a built-in whitelist in the Server side DNS server instead of external DNSBL and DNSWL. As an OpenFlow Controller, we used Trema [6] along with MySQL [7] for the temporary whitelist. As an OpenFlow Switch, we used Open vSwitch [8]. The server side DNS server was built with Perl module “Net::DNS::Nameserver” [9].

3. 2 Functionality evaluation of the prototype system

We performed functionality evaluation of the prototype system at OU. The specifications of the components are shown in Table 1. In this evaluation, only the IP address of Client 1 was registered in WL on the Server side DNS server. Therefore, all the packets of Client 1’s flows were forwarded to Server 1 while all the packets of Clients 2 and 3’s flows were forwarded to Server 2. We had Clients 1 and 2 send ICMP echo packets to Server 1. Accordingly, those packets sent from Clients 1 and 2 were forwarded to Server 1 and 2, respectively.

After sending ICMP echo packets, flow entries and data paths were created in the Open vSwitch, as shown in Figures 4 and 5, respectively. According to these results, we confirmed that the prototype system at OU worked well as we expected.

Table 1 The specifications of the componets (OU).

Host type	CPU/Main Memory	Running OS
Server side DNS server	Xeon E5620 2.40GHz/1GB	FreeBSD 8.2-RELESE
Client side DNS server	Xeon E5620 2.40GHz/1GB	FreeBSD 8.2-RELESE
OpenFlow Switch	Core 6700 2.66GHz/2GB	Ubuntu 12.04-Release
OpenFlow Controller	Core i5-4440 3.10GHz/8GB	Ubuntu 12.04-Release
Clients 1-3	Xeon E5620 2.40GHz/1GB	FreeBSD 8.2-RELESE
Server 1	Core i3-2100 3.10GHz/2GB	FreeBSD 8.2-RELESE
Server 2	Core 2 6300 2.66GHz/2GB	Ubuntu 10.04-Release

```
OpenVSwitch:~$ sudo ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x5,duration=3.474s,table=0,n_packets=0,n_bytes=0,priority=65
535,ip,nw_src=150.46.47.121,nw_dst=192.168.1.5 actions=output:2
cookie=0x6,duration=3.474s,table=0,n_packets=0,n_bytes=0,priority=20
0,ip,in_port=2 actions=output:1
cookie=0x7,duration=3.474s,table=0,n_packets=0,n_bytes=0,priority=0
actions=mod _nw_dst:192.168.1.4,output:3
```

Figure 4 The flow entries of the Open vSwitch (OU).

```
OpenVSwitch:~$ sudo ovs-dpctl dump-flows br0
in_port(1),eth(src=00:0d:28:66:55:00,dst=00:30:67:e3:1d:ba),eth_type(0x
0800),ipv4(src=150.46.47.121,dst=192.168.1.5,proto=1,tos=0,ttl=63,frag
=no),icmp(type=8,code=0),packets:6,bytes:588,used:0.088s,actions:2
in_port(2),eth(src=00:30:67:e3:1d:ba,dst=00:0d:28:66:55:00),eth_type(0x
0800),ipv4(src=192.168.1.5,dst=150.46.47.121,proto=1,tos=0,ttl=64,frag
=no),icmp(type=0,code=0),packets:6,bytes:588,used:0.088s,actions:1
in_port(1),eth(src=00:0d:28:66:55:00,00:30:67:e3:1d:ba),eth_typ(0x0800),
ipv4(src=150.46.47.129,dst=192.168.1.5,proto=1,tos=0,ttl=63,frag=no),ic
mp(type=8,code=0),packets:6,bytes:588,used:0.088s,actions:set(ipv4(sr
c=150.46.47.129,dst=192.168.1.4,prot=1,tos=0,ttl=63,frag=no)),2
```

Figure 5 The data paths of the Open vSwitch (OU).

3. 3 Performance evaluation of the prototype system

Then, we evaluated the performance of the prototype system at TUAT. The layout of the prototype system was almost the same as shown in Figure 3, except for absence of both DNS servers. Instead of WL in Server side DNS server, only the IP address of Client 1 was registered in MySQL of OpenFlow Controller in advance. Therefore, all the packets of Client 1's flows were forwarded to Server 1 while all the packets of Clients 2 and 3's flows were forwarded to Server 2. Since we assumed the throughput of real firewalls is not so high compared with that of the network, we reduced the bandwidths of the links to Servers to 100Mbps while the bandwidths of other links were 1Gbps. The specifications of the components are shown in Table 2.

In order to measure the throuput between Client 1 and Server1, we used "iPerf" [10] for 10 seconds and calculated the average throuputs over 5 measurements. To generate attack traffic, we ran "hping3" [11] on Clients 2 and 3 with "--faster" or "--flood" options, which mean sending ICMP packets "100 per second" or "as fast as possible," respectively. For comparison, we also measured the throuputs in some system configurations where "an L2SW was used instead of Open vSwitch and all traffic was forwarded to Server1" (L2SW), "Open vSwitch was used and and all traffic was forwarded to Server 1" (All to Server1), and "attack traffic was dropped" (Attack dropped), as well as "attack traffic was forwarded to Server 2" (Attack to Server2). The result of the performance evaluation is shown in Figure 6.

Table 2 The specifications of the componets (TUAT).

Host type	CPU/Main Memory	Running OS
OpenFlow Switch	Pentium 1403 v2 2.60GHz/8GB	Debian 8.3
OpenFlow Controller	Xeon E31245 3.30GHz/8GB	Debian 8.3
Clients 1-3, Servers 1-2	Core 2 DUO 2.66GHz/2GB	Debian 8.3

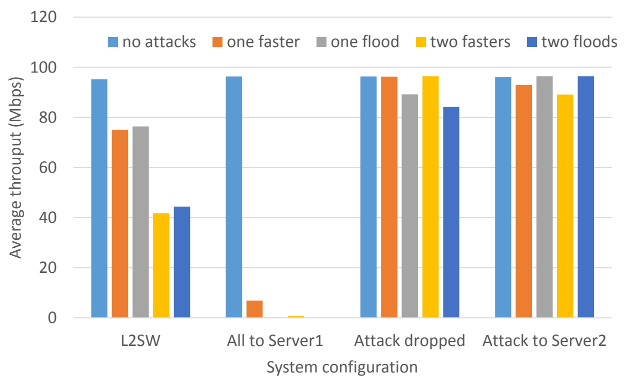

Figure 6 The average throuputs against kinds of attacks in various system configurations (TUAT).

According to this figure, we can find that throuputs of "L2SW" and "All to Server1" were considerably reduced due to attacks, while throughputs of "Attack dropped" and "Attack to Server2" were high enough even under heavy attacks. Consequently, we have confirmed that although the overhead of OpenFlow Switch is not negligible compared with that of L2SW, the proposed proactive firewall system can protect the traffic of trusted hosts from other traffic.

4. Conclusions

In this paper, we proposed a proactive firewall system in cooperation with SDN and DNS. The proposed system introduces the client-subnet option of EDNS0 to obtain the peer IP address in advance and protects traffic of the peer from other traffic using SDN technogoly. Through the funcationality evaluation and the performance evaluation, we confirmed the prototype system worked well as we expected and performed good throughput. The future works include the evaluations of the entire system with the real firewall appliances.

Acknowledgement

This work was partially supported by JSPS Grants-in-Aid for Scientific Research (KAKENHI) Grant Number JP25330105.

References

[1] S. Y. Yoon, B. K. Kim, J. T. Oh and J. S. Jang, "High Performance Session State Management Scheme for Stateful Packet Inspection", Managing Next Generation Networks and Services, Lecture Notes in Computer Science, Vol.4773, pp.591-594, 2007.

[2] Y. H. Cho, W. H. Mangione-Smith, "Deep Packet Filter with Dedicated Logic and Read Only Memories," Field Prog. Logic and Applications, Aug. 2004, pp. 125-134.

[3] C. Contavalli, W. van der Gaast, D. Lawrence and W. Kumari: "Client Subnet in DNS Queries," RFC7871, IETF, 2016.

[4] MaxMind, Inc., "GeoIP® Databases & Services: Industry Leading IP Intelligence | MaxMind" (online), available at <https://www.maxmind.com/en/geoip2-services-and-databases> (accessed 2016-05-25).

[5] T. Otsuka, Gada, N. Yamai, K. Okayama and Y. Jin, "Design and Implementation of Client IP Notification Feature on DNS for Proactive Firewall System," Proc.of 2015 IEEE 39th International Conference on Computer Software and Applications (COMPSAC 2015) Workshops, pp.127-172, 2015.

[6] Trema, "Trema : Full-Stack OpenFlow Framework in Ruby and C" (online), available at <http://trema.github.io/trema/> (accessed 2016-05-25).

[7] Oracle Corporation and/or its affiliates, "MySQL" (online), available at <http://www.mysql.com/> (accessed 2016-05-25)

[8] Open vSwitch, "Open vSwitch" (online), available at <http://openvswitch.org/> (accessed 2016-05-25).

[9] M. Fuhr, C. Reinhardt, R. Martin-Legene, and O.M. Kolkman, "Net::DNS::Nameserver" (online), available at <http://search.cpan.org/dist/Net-DNS/lib/Net/DNS/Nameserver.pm> (accessed 2016-05-25).

[10] V. Gueant, "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool" (online), available at <https://iperf.fr/> (accessed 2016-05-25).

[11] S. Sanfilippo, "Hping - Active Network Security Tool" (online), available at <http://www.hping.org/> (accessed 2016-05-25).