Seo and Lee *SpringerPlus (2016)5:1878*
DOI 10.1186/s40064-016-3569-3

SpringerPlus

RESEARCH Open Access

A study on efficient detection of network-based IP spoofing DDoS and malware-infected Systems

Jung Woo Seo* and Sang Jin Lee

*Correspondence:
korea002@korea.ac.kr
Graduate School
of Information Security, Korea
University, 145 Anam-ro,
Seongbuk-gu, Seoul, Korea

Abstract

Large-scale network environments require effective detection and response methods against DDoS attacks. Depending on the advancement of IT infrastructure such as the server or network equipment, DDoS attack traffic arising from a few malware-infected systems capable of crippling the organization's internal network has become a significant threat. This study calculates the frequency of network-based packet attributes and analyzes the anomalies of the attributes in order to detect IP-spoofed DDoS attacks. Also, a method is proposed for the effective detection of malware infection systems triggering IP-spoofed DDoS attacks on an edge network. Detection accuracy and performance of the collected real-time traffic on a core network is analyzed thru the use of the proposed algorithm, and a prototype was developed to evaluate the performance of the algorithm. As a result, DDoS attacks on the internal network were detected in real-time and whether or not IP addresses were spoofed was confirmed. Detecting hosts infected by malware in real-time allowed the execution of intrusion responses before stoppage of the internal network caused by large-scale attack traffic.

Introduction

Hardware performance has seen astonishing advancements with the improving environment of information services. The network infrastructure available today is capable of sending gigabytes of data in just a few seconds. The downside of such developments is that information systems and network infrastructure are more prone to malware-induced cyber-attacks. According a recent report on malware, Linux-based botnets account for the highest proportion of DDoS attacks at 45%, and they pose major threats to the information services of companies when installed in embedded devices such as Wi-Fi, routers, and NAS (Ferguson and Senie 2000; Baker and Savola 2004).

According to a report by Akamai Technologies (Akamai 2015), Linux-based XOR DDoS malware is able to launch DDoS attacks of up to 150 Gbps, and malware-infected systems can attack an average of 20 websites per day. To install and execute malware, attackers acquire the administrative rights of a vulnerable system and run a Shell command in order to install a malware program containing a rootkit function, which allows them to hide their presence.

As demonstrated above, the development of IT has resulted in threats unseen in the past and corporate information service environments are at a greater risk than ever. XOR DDoS malware, one of the most threatening types of malware that can infect Linux-based systems, launches massive IP-spoofed DDoS attacks that paralyze internal networks. Companies have selected Linux-based systems over the more vulnerable Windows operating system as a method of security enforcement, but the various techniques and tools developed by attackers are posing significant threats to businesses.

Past research has focused on the detection of outbound-to-inbound DDoS attacks (Duan et al. 2008; Wang et al. 2007; Feinstein et al. 2003) or the detection of attacks by analyzing information retrieved from botnet agents in internal systems (François et al. 2011; Abu Rajab et al. 2006). However, few studies exist on the detection of network-based attacks when the internal network is under massive volumes of DDoS traffic caused by the IP-spoofed DDoS malware or other infections. For instance, when a DDoS attack occurs due to a malware-infected system in the Demilitarized Zone (DMZ), SYN flooding takes place in the network section from the host system to the security system. The depletion of network resources disrupts network services in the corresponding bandwidth. Rapid detection and response to malware-infected systems is the most effective way of ensuring the availability of the internal network (Perdisci et al. 2013; Beverly et al. 2009; Bremler-Barr and Levy 2005).

This paper applies the DDoS malware finder (DMF) algorithm in order to detect abnormalities when the host of an internal network is infected with DDoS malware based on Spoofed IP Address and launches massive volumes of DDoS attacks. The DMF algorithm derives patterns from feature information of DDoS malware based on Spoofed IP Address and applies the matching rule to enable real-time detection. Malware-infected systems are identified with reference to the DMF table.

The rest of this paper is organized as follows: second section describes the background of the study and related work, third section contains the motivation and challenge, fourth section gives the system overview, fifth section presents the system model, and sixth section is the evaluation. The last section is the conclusion.

Background and related work

Background

Malware Must Die team members first detected XOR DDoS in September 2014. A Trojan malware was used to hijack Linux machines in order to build a botnet for DDoS (Akamai 2015).

The bandwidth of DDoS attacks coming from the XOR DDoS botnet has ranged from a few gigabits per second to 150Gbps. The botnet has attacked up to 20 targets per day, 90% of which are in Asia. Two DDoS attacks were caused by the XOR DDoS botnet on the weekend of August 22–23. One of the attacks measured nearly 50 Gbps and the other reached nearly 100 Gbps. XOR DDoS is an example of attackers building botnets from Linux systems instead of Windows-based machines. Other recent examples of Linux-based malware include the Spike DDoS toolkit and IptabLes and IptabLex malware. There is an increasing number of Linux vulnerabilities for malicious actors to target, such as the heap-based buffer overflow vulnerability found earlier this year in

the GNU C Library. However, XOR DDoS itself does not exploit a specific vulnerability (Arbor Networks 2015; Akamai 2015).

Related work

Although a significant amount of literature has been produced on botnet detection, botnet detection approaches using flow analysis techniques have only emerged in the last few years.

Zhao et al. (2013) proposed a new approach to detect botnet activity based on traffic behavior analysis by classifying network traffic behavior using machine learning. Traffic behavior analysis methods do not depend on the packets payload, which means that they can work with encrypted network communication protocols. Network traffic information can usually be easily retrieved from various network devices without affecting significantly network performance or service availability. The proposed approach detects botnet activity by classifying behavior based on time intervals.

BotHunter (John and Tafvelin 2007) consists of intrusion detection system (IDS) components, used to observe inbound and outbound traffic flow, and a dialog correlation engine that generates the flow of bot infections. The two BotHunter plugins are the Statistical sCan Anomaly Detection Engine (SCADE) and the Statistical payLoad Anomaly Detection Engine (SLADE).

SCADE performs inbound scan detection and outbound scan detection by categorizing anomalies as high-severity (HS) or low-severity (LS). SLADE, based on a byte-distribution payload detection technique, inspects the payloads of all packets sent by the service being monitored and provides warnings when the N-gram frequency exceeds the normal range (Gu et al. 2008).

While BotHunter offers a remote repository for users to evaluate bot activities and collect information, it is difficult to detect anomalies in bots that use encrypted channels in order to communicate with the Command and Control (C&C) server and stealth-scanning bots. Since bot infection is determined based on the behavioral patterns of modified bots, bots with signatures updated from variations in traffic patterns are also not easy to identify. To resolve these issues, it is necessary for the server to automatically collect and analyze patterns, as well as to constantly renew them (Tegeler et al. 2012; Gu et al. 2008).

Zeidanloo et al. (2010) proposed a botnet detection approach based on the monitoring of network traffic characteristics in a similar way to BotMiner. In their work, a three stages process of filtering, malicious activity detection and traffic monitoring is used to group bots by their group behavior. The proposed approach divides the concept of flows into time periods of 6 h and clusters these flow intervals with known malicious activity. The effects of different flow interval durations were not presented, and the accuracy of the approach is unknown (Zhao et al. 2013).

Wang et al. (2009) presented a detection approach of peer-to-peer based botnets by observing the stability of control flows in initial time intervals of 10 min. They developed an algorithm which measures the stability of flows and exploits the property that bots exhibit similar behavior in their command search and perform these tasks independently of each other and frequently. They show that by varying parameters in their algorithm,

they were able to classify 98% of Storm C&C data as stable, though a large percentage of non-malicious peer-to-peer traffic were also classified as such (Zhao et al. 2013).

Motivation and challenge

Recently, there has been an increase in attackers gaining the administrative rights of Linux-based systems by exploiting web service vulnerabilities such as SQL Injection, OpenSSL, and file uploads. The attackers access the C&C server or malware distribution site with their administrative status, and execute the wget or curl command in order to perform a forced installation of the downloaded malware. Figure 1 shows the installation of the malware program in the/etc./rc.3d and/etc./rc5.d directories after a reboot of a system infected with malware. The installed malware communicates with the C&C server and prepares to launch attacks (Sakib and Huang 2016; Park and Lee 2001).

Table 1 is an example of a SYN flooding payload caused by an IP-spoofed DDoS attack in an actual information service environment.

The SYN flooding attacks are launched by the infected system on an external, unspecified target and IP spoofing makes it more difficult to detect the source of the infection. If the infected system behind the attacks cannot be efficiently identified, the bandwidth of the internal network becomes consumed by DDoS attacks, resulting in a disruption of network services (Liu and Bi 2015; Freiling et al. 2005). Figure 2 shows the detection of DDoS attacks generated by a system infected with malware.

Figure 3 shows the monitoring of the Network Management System (NMS) when infected by IP-spoofed DDoS malware. The screen shows the DMZ network, which provides services with an average traffic of 600 Mbps. As shown, the infected system

```
lrwxrwxrwx.  1 root root   16 2015-05-15 09:19 S60vsftpd -> ../init.d/vsftpd
lrwxrwxrwx   1 root root   18 2015-12-16 17:33 S80sendmail -> ../init.d/sendmail
lrwxrwxrwx.  1 root root   19 2015-05-15 11:28 S82abrt-ccpp -> ../init.d/abrt-ccpp
lrwxrwxrwx.  1 root root   19 2015-05-15 11:28 S82abrt-oops -> ../init.d/abrt-oops
lrwxrwxrwx.  1 root root   15 2015-05-14 18:22 S82abrtd -> ../init.d/abrtd
lrwxrwxrwx   1 root root   15 2015-05-20 10:00 S85httpd -> ../init.d/httpd
lrwxrwxrwx   1 root root   20 2015-06-15 08:30 S90axzfxbozli -> ../init.d/axzfxbozli
lrwxrwxrwx.  1 root root   15 2015-05-14 18:27 S90crond -> ../init.d/crond
lrwxrwxrwx   1 root root   20 2015-11-05 17:09 S90eoikyuczay -> ../init.d/eoikyuczay
lrwxrwxrwx   1 root root   20 2015-12-24 10:37 S90fdhwxilkdu -> ../init.d/fdhwxilkdu
lrwxrwxrwx   1 root root   20 2015-06-12 08:29 S90fnnqoipzzq -> ../init.d/fnnqoipzzq
lrwxrwxrwx   1 root root   20 2015-05-31 08:29 S90fnwonfblyj -> ../init.d/fnwonfblyj
lrwxrwxrwx   1 root root   16 2015-05-15 13:53 S90oracle -> ../init.d/oracle
lrwxrwxrwx   1 root root   20 2015-06-30 08:30 S90pqztdgasrr -> ../init.d/pqztdgasrr
lrwxrwxrwx   1 root root   20 2015-12-24 10:37 S90uuiemvfoxx -> ../init.d/uuiemvfoxx
lrwxrwxrwx   1 root root   20 2015-08-11 18:09 S90ycjmhmutth -> ../init.d/ycjmhmutth
lrwxrwxrwx   1 root root   18 2015-06-19 15:57 S92fail2ban -> ../init.d/fail2ban
lrwxrwxrwx.  1 root root   13 2015-05-14 18:21 S95atd -> ../init.d/atd
```

Fig. 1 Installation of XOR DDoS malware

Table 1 Capturing network traffic by XOR DDoS

06:05:24.260515 IP x.x.x.x.7318 > y.y.y.y.80: Flags [S], seq 479609867:479610763,win65535,length896
06:05:24.260540 IP x.x.x.x.2104 > y.y.y.y.80: Flags [S], seq 137948748:137949644,win65535,length896
06:05:24.260560 IP x.x.x.x.58852 > y.y.y.y.80: Flags [S], seq 3856952941:3856953837,win65535,length 896
…
06:05:24.260574 IP x.x.x.x.4375 > y.y.y.y.80: Flags [S], seq 286734425:286735321,win65535,length896
06:05:24.260583 IP x.x.x.x.62129 > y.y.y.y.80: Flags [SE], seq 4071711351:4071712247,win65535,length896

tcp syn flooding	2016/01/19 11:17:45	23,27,66,73	TCP/5761	59,36,97,66	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,102	TCP/46000	59,36,97,21	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,107	TCP/63500	59,36,97,66	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,248	TCP/4189	59,36,97,46	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,174	TCP/17611	59,36,97,24	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,104	TCP/63449	59,36,97,62	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,186	TCP/37897	59,36,97,62	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,196	TCP/51217	59,36,97,46	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,216	TCP/49909	59,36,97,21	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,66	TCP/64213	59,36,97,46	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,115	TCP/5997	59,36,97,66	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,233	TCP/23442	59,36,97,66	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,206	TCP/34002	59,36,97,24	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,218	TCP/33635	59,36,97,46	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,136	TCP/26769	59,36,97,21	TCP/3389
tcp syn flooding	2016/01/19 11:17:45	23,27,66,88	TCP/49621	59,36,97,46	TCP/3389

Fig. 2 DDoS attacks detected by intrusion detection system

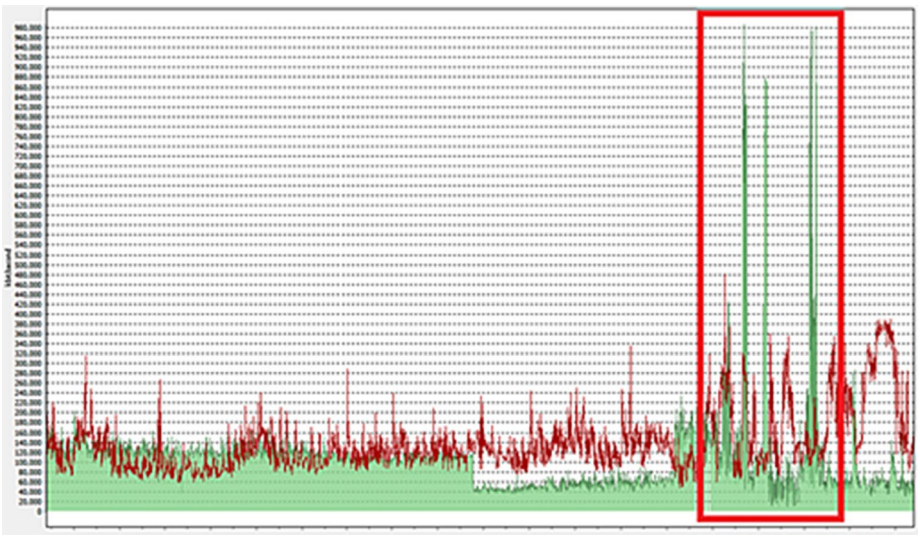

Fig. 3 Monitoring of network traffic by XOR DDoS attack

generated DDoS traffic of 2 Gbps. The unexpected increase in traffic, exceeding the 1 Gbps limit, caused the failure of the entire network.

Ultimately, the most effective way of ensuring network availability is the rapid detection and response to systems that generate huge volumes of traffic after being infected with IP-spoofed DDoS malware. This is also related to the reliability of information services.

One method of preventing malware infection is to install vaccine programs for all Linux-based systems, but this is not the best solution for companies. Installing vaccines on all Linux systems is not only expensive, but also requires the software license to be renewed each year.

Another method is to install anti-bot programs to detect malicious bots in the local system. However, anti-bot programs may be impossible to install depending on hardware performance and additional licenses have to be purchased if new hosts are added. As illustrated in Fig. 4, it is important to consider a network-based approach to malware detection in order to overcome the aforementioned issues.

By analyzing an actual system infected with IP-spoofed DDoS malware, the following features were identified. First, massive volumes of SYN flooding attacks were generated by creating more than two million SYN packets per minute. Second, IP spoofing was performed in order to conceal the source of the attacks. The list of attacked systems was downloaded from the C&C server and periodically updated for the next series of attacks. Figure 5 shows the IP address and port number of IP-spoofed DDoS attacks.

This study proposes a method that can detect network-based IP-spoofed DDoS attacks and efficiently identify malware-infected systems.

System overview

The DMF algorithm analyzes the feature information of the traffic header in order to detect IP-spoofed DDoS malware. The three phases are extraction, analysis, and detection. In the first phase, the attack features of the IP-spoofed DDoS malware are analyzed after categorizing the internal network by service area and collecting real-time network traffic. Based on the Layer 3 switch, network services are categorized into user

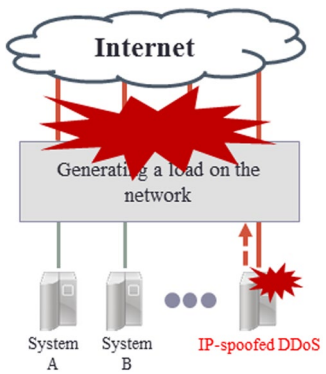

Fig. 4 DDoS detection based on network

EVENT_TIME	S_INFO	S_PORT	D_INFO	D_PORT	PROTOCOL
2016.02.01 11:11:34	223.47.66.212	31373	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.164	30659	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.240	56993	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.160	44365	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.102	37074	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.100	12088	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.224	41986	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.92	24379	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.136	42746	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.196	28977	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.148	33022	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.136	52189	59.36.97.11	80	TCP
2016.02.01 11:11:34	223.47.66.176	23760	59.36.97.11	80	TCP

Fig. 5 IP address and port number of IP-Spoofed DDoS attacks

bandwidth, server farm bandwidth, and branch office. Traffic collection is configured in port mirroring and a DMF table is created by extracting feature information, including the IP address and MAC address of the Ethernet headers, the current time, and the time interval. In Table 2, the columns specify the attribute information of a DMF table.

In the second phase, the proposed detection algorithm is applied and statistical features are analyzed in order to detect IP-spoofed DDoS malware. The extracted IP addresses and MAC addresses from Ethernet headers of real-time traffic are compared with the attribute information provided in the DMF table to determine if IP-spoofed DDoS malware infection has occurred. Figure 6 shows the process for detection of IP-spoofed DDoS malware.

The DMF table consists of the attribute information of traffic headers. The time interval is calculated if there is a match between the IP addresses and MAC addresses of real-time traffic with the property values in the DMF DB. Time-interval values are used to check for IP-spoofed DDoS malware infection.

Instead of referring to traffic payload, the proposed algorithm relies on the feature information of the IP-spoofed DDoS malware for malware detection.

System model

The purpose of the experiment is to detect malware-infected systems after detecting IP-spoofed DDoS based on an analysis of real-time traffic. This section examines the system model for the proposed method.

Gathering on the network traffic

To implement the DMF algorithm, port mirroring is configured at the Layer 3 switch. The reason for the port mirroring configuration is to detect malware infection based on an analysis of traffic routed through the network equipment and to identify infected systems. Figure 7 shows the schematic diagram for real-time network traffic collection.

Table 2 Attribute information of a DMF table

Attribute	Type	Feature data	Description
SN	VARCHAR(20)	S00001	Sequence number to traffic
SGN	VARCHAR(20)	G00001	Same group number with the same destination IP
Attribute Info.			
SRCIP	VARCHAR(15)	58.203.201.36	Source IP
SRCPORT	VARCHAR(6)	5312	Source Port
DSTIP	VARCHAR(15)	211.106.66.102	Destination IP
DSTPORT	VARCHAR(6)	80	Destination Port
MAC	VARCHAR(17)	D0-27-88-47-15-4B	Media access control (MAC)
PT	VARCHAR(8)	TCP	Protocol
AT	DATE	2016.08.12 07:05:28	Destination IP connection time
TIN	FLOAT	2 s	Interval from previous traffic
CND	INTEGER	28 hit	Destination IP connection hits
RATD	FLOAT	136 s	Interval from previous traffic connected to destination IP
CNTI	INTEGER	12 hit	Hits to the destination IP over 3 min
STATE	VARCHAR(20)	Abnormal	Traffic anomaly status

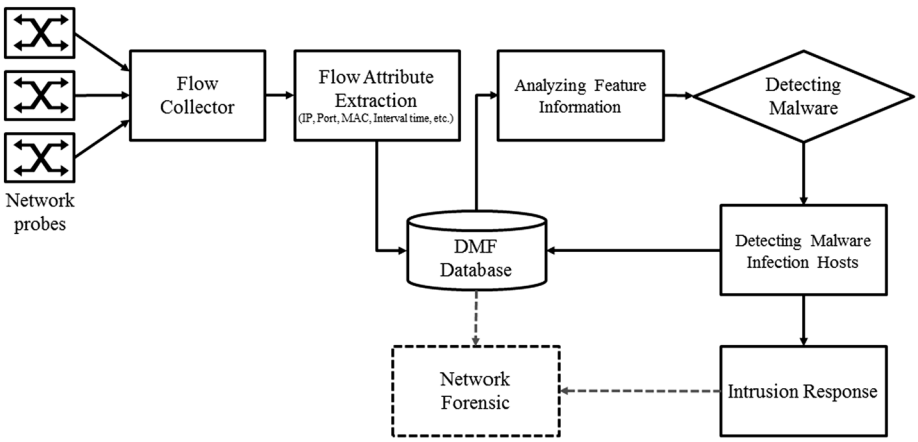

Fig. 6 Process for detection of IP-spoofed DDoS malware

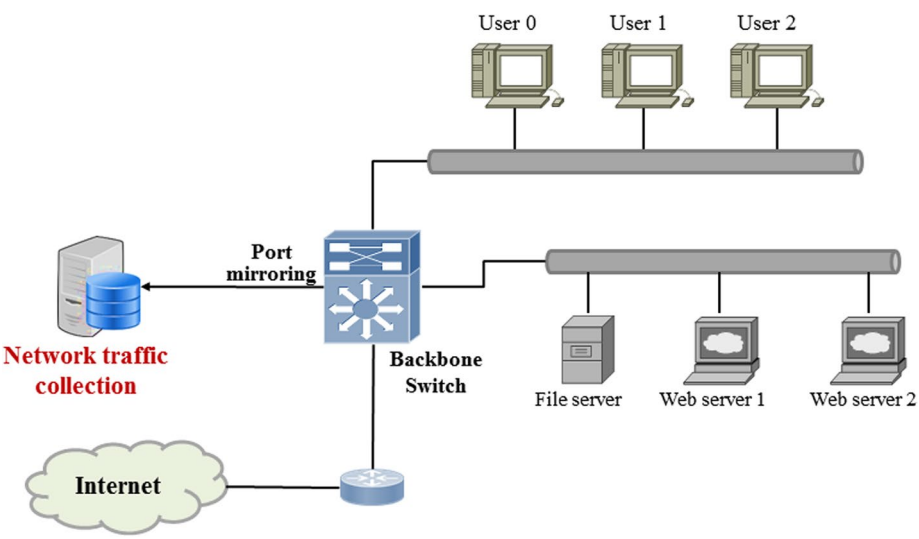

Fig. 7 Collection and analysis of network packets

Extraction on the feature information

As shown in Fig. 7, the attribute information by TCP/IP and Ethernet header are extracted from the collected traffic and used as basic data in generating the DMF DB. Figure 8 shows the protocol stack used in extracting the feature information from real-time traffic.

As shown in Fig. 8, the headers of traffic flowing out of the internal network are used to extract feature information. Figure 9 shows the process of collecting SYN packets sent by the server farm and extracting the feature information from the collected traffic.

Creation of the DMF table

The DMF table is used to detect DDoS attacks in real-time traffic and contains feature values that are useful in identifying malware-infected systems. The procedures for generating the DMF table are as follows:

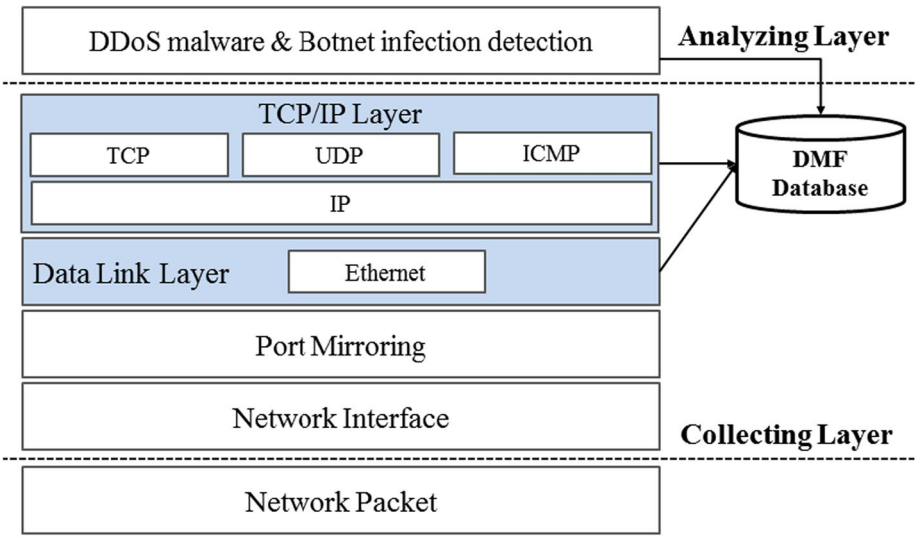

Fig. 8 Protocol stack for extraction of feature information

- The SN field of the DMF table represents the traffic sequence and the SGN field means the number of the group connecting to the same destination IP.
- The traffic header IP address and MAC address is extracted and stored in the SRCIP, DSTIP, and MAC fields for the DMF table. The communication protocol between the two hosts and the access time are also stored.
- The TIN field represents the interval between previous traffic and current traffic.
- The CND field represents the connection hits to the destination IP and the RATD field represents the reconnect interval of the destination IP.
- The STATE field is categorized into two types. If the TIN field value and RATD value are zero and the CNTI field divided by 180 s is ≤ 1 then the field value is set as "abnormal". All other instances are set as "normal". Table 3 shows the algorithm for creating the DMF table.

Figure 10 shows an example of the data stored in the DMF table.

Analysis of feature information

Malware-infected hosts receive attack commands from the C&C server or leaks important data to the C&C server. Also, they receive DDoS attack commands from the C&C server or generate large scale IP-spoofed DDoS attack traffic at a specified time. According to the performance of the host and the network bandwidth, the DDoS traffic generated at this time can cripple the internal network and makes it extremely difficult to detect IP-spoofed malware-infected hosts. The proposed method to detect the DDoS attack is to use the DMF table in order to analyze the traffic generation time and generation frequency, thereby determining if it is IP-spoofed by using the IP address and MAC address. To detect malware-infected hosts, the connection times to the MAC address and destination IP address and connection frequency are used. Figure 11 shows a DMF table with the analysis results of the attribute information of the network traffic.

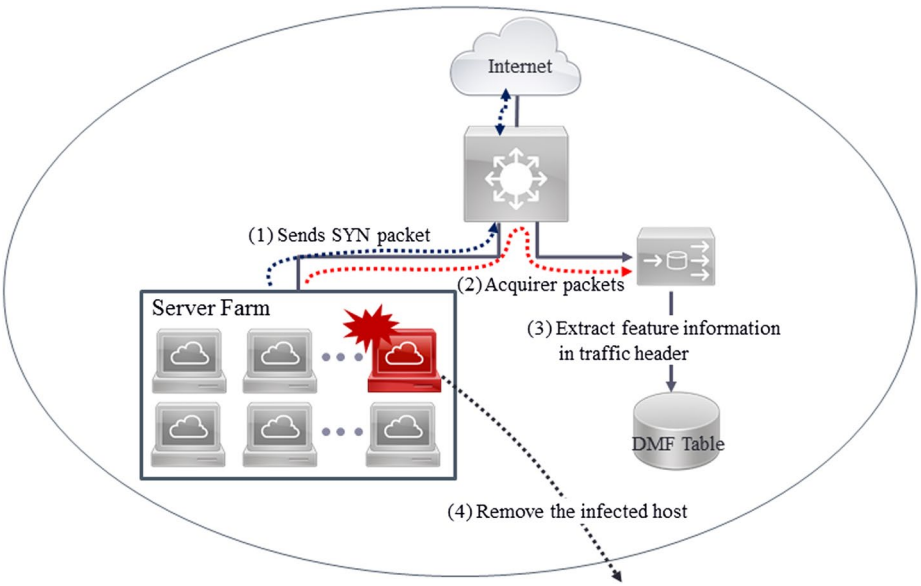

Fig. 9 Extraction of feature information

Detection of IP-spoofed DDoS attack

In order to detect IP-spoofed DDoS attacks, attribute information collected from the core network is analyzed. Attribute information is stored in the DMF table and, as seen in Fig. 11, the attribute's relative information is used to detect attacks. Table 4 shows the algorithm to detect IP-spoofed DDoS attacks.

Figure 12 shows the timeline of traffic information generated on a network when a malware-infected host executes an IP-spoofed DDoS attack.

As shown in Fig. 12, analysis results of the attributes of IP-spoofed DDoS malware showed that it was periodically communicating with the C&C server and the attack target was downloaded from the C&C server and updated. The IP-spoofed DDoS attack was powerful enough to cause an outage to the core network in a gigabit Ethernet environment.

Finding of a system infected by IP-spoofed DDoS malware

Upon detecting the IP-spoofed DDoS attack with the proposed methodology, it is necessary to find the malware-infected host and eliminate the cause. Because of IP address spoofing, it is impossible to block a particular IP address in the firewall, and even if we were to block a range of IP addresses, the internal network would either come to a halt or have extreme delays as a result of the large volume traffic generated by the malware. Therefore, it is necessary to quickly quarantine the host that is generating the DDoS attacks to an edge network.

IP-spoofed DDoS attacks create tampered IP address packets on a host infected by malicious code and utilizes the maximum performance of the host to generate large volume traffic. According to the DDoS attack, network resources are consumed and because the web services and intranet services are not executed normally, resources are depleted as time goes by. Table 5 shows an algorithm to effectively detect malware-infected hosts.

Table 3 Creation of DMF table

Algorithm 1 : Creation of DMF Table

Input *: T = { P_{srcip}, $P_{srcport}$, P_{dstip}, $P_{dstport}$, P_{mac}, $P_{protocol}$, $P_{accesstime}$ } where T is the attribute value about traffic header*
Output *: DMF Table*

```
begin
    pval[ ][ ] ← null
    T ← EntryPoint(P)
    mval ← 180          //reference value for ddos detection
    while r is not the end of T do
        i ← Ø
        pval[i][0] ← "P" ∪ rownum
        if { P_dstip ∈ pval and pval ≠ null } then   //The same desIP is included in the DMF table
            pval[i][1] ← pval[x][1]                  //x is a array number detected by the searching
            //If the same destination IP is included in the DMF table, [x] is the address of the array
        else
            pval[i][1] ← "G" ∪ rownum
        end
        pval[i][2] ← P_srcip                         //Source IP
        pval[i][3] ← P_srcport                       //Source Port
        pval[i][4] ← P_dstip                         //Destination IP
        pval[i][5] ← P_dstport                       //Destination Port
        pval[i][6] ← P_mac                           //MAC(Media Access Control) address
        pval[i][7] ← P_protocol                      //Protocol
        pval[i][8] ← P_accesstime                    //Access Time
        pval[i][9] ← P_accesstime - pval[i-1][9]     //Time interval between the traffic
        if { P_dstip ∈ pval and pval ≠ null } then
            pval[i][10] ← pval[x][10] + 1
            //CND is a number of times to reconnect to the destination IP
            pval[i][11] ← P_accesstime - pval[x][8]
            //RATD is time interval to reconnect to the destination IP
            j ← length of pval array
            foreach { (currentTime() - pval[j][8]) < mval } do
                cnt ← cnt + 1
            end
            pval[i][12] ← cnt
            //CNTI is a number of times to reconnect destination IP during 3minute
        else
            pval[i][10] ← 0
            pval[i][11] ← 0
        end
        tinVal ← pval[i][9]
        cndVal ← pval[i][10]
        cntiVal ← pval[i][12]

        if { tinVal is zero and ratdVal is zero and (mval / cntiVa1) ≤ 1 } then
          //when the average time interval is less than 1 for 3 minutes
            pval[i][13] ← "abnormal"          //ip-spoofed ddos attack
        else
            pval[i][13] ← "normal"            //normal traffic
        end
        i ← i + 1
    end
    return DMF Table ← pval
end
```

Intrusion response on infected system

When a DDoS attack manifests in the internal network, it is necessary to detect the intruding system and respond before it consumes all of the internal network's bandwidth. Also, it is necessary to understand the intrusion cause and secure the audit trail

SN	SGN	SRCIP	SRCPORT	DSTIP	DSTPORT	MAC	PT	AT	TIN	RNDS	RIDS	RNTI	STATE
S001023	G000203	23.23.135.172	20301	116.67.75.149	80	C0-3F-D5-AC-10-A6	TCP	2016-03-29 17:53:11	0	1	0	1	normal
S001024	G000204	23.20.205.42	35091	125.60.33.45	80	1E-3F-51-56-C0-3F	TCP	2016-03-29 17:53:11	0	1	0	1	normal
S001025	G000203	23.23.135.172	20301	116.67.75.149	80	C0-3F-D5-AC-10-A6	TCP	2016-03-29 17:53:12	1	2	1	2	normal
S001026	G000205	23.24.135.193	14390	113.29.189.132	80	D9-01-G0-E2-90-MO	TCP	2016-03-29 17:53:12	0	1	0	1	normal
S001027	G000203	23.20.206.172	20301	211.115.106.76	80	C0-3F-D5-AC-10-A6	TCP	2016-03-29 17:53:14	2	3	2	3	normal
S001028	G000206	23.19.135.31	10219	211.244.82.218	80	00-50-56-C0-00-08	TCP	2016-03-29 17:53:15	1	1	0	1	normal
S001029	G000207	23.19.135.145	34090	61.247.193.200	80	1E-3F-51-56-C0-3F	TCP	2016-03-29 17:53:15	0	1	0	1	normal
S001030	G000205	23.24.135.193	14390	117.52.2.19	80	D9-01-G0-E2-90-MO	TCP	2016-03-29 17:53:15	0	2	3	2	normal
S001031	G000205	23.24.135.193	14390	117.52.2.19	80	D9-01-G0-E2-90-MO	TCP	2016-03-29 17:53:16	1	3	1	3	normal
S001032	G000205	23.24.135.193	14390	113.29.189.132	80	D9-01-G0-E2-90-MO	TCP	2016-03-29 17:53:20	4	4	4	4	normal
S001033	G000208	23.27.136.220	19021	117.52.90.77	80	00-01-1E-3F-51-56	TCP	2016-03-29 17:53:20	0	1	0	1	normal
S001034	G000205	23.24.135.193	14390	114.108.157.13	80	D9-01-G0-E2-90-MO	TCP	2016-03-29 17:53:22	2	5	2	5	normal
S001035	G000209	23.23.149.81	20901	58.229.187.44	80	Q2-87-AE-C0-EF-80	TCP	2016-03-29 17:53:22	0	1	0	1	normal
S001036	G000203	23.23.135.172	20301	116.67.75.149	80	C0-3F-D5-AC-10-A6	TCP	2016-03-29 17:53:22	0	4	8	4	normal
S001037	G000210	23.20.202.33	33021	101.79.245.37	80	C0-TU-08-87-AE-C0	TCP	2016-03-29 17:53:25	3	1	0	1	normal
S001038	G000211	23.20.201.50	23090	211.115.106.76	80	32-OP-ET-FD-O0-87	TCP	2016-03-29 17:53:25	0	1	0	1	normal

Fig. 10 Feature information of collected traffic

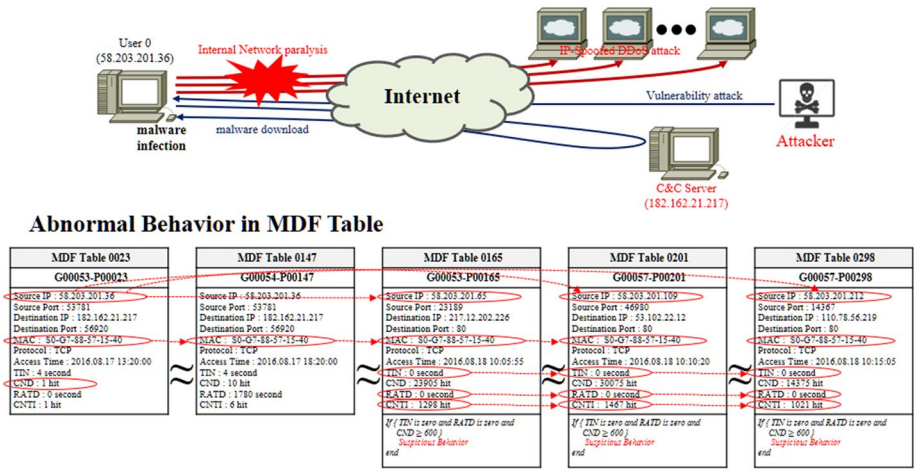

Fig. 11 Analysis of feature information using DMF table

in order to eliminate any weak spots. The proposed method is for network forensics to understand the traffic flow and to use the attribute values of the DMF table to analyze the malware-infected system and its attack tendencies. Figure 13 shows the network forensics execution plan using the DMF table.

Evaluation

The testbed environment is organized as shown in Fig. 14 in order to evaluate the efficiency of the proposed algorithm, detect IP-spoofed DDoS attacks, and identify malware-infected systems. This section describes the differences between the proposed method and existing technology and evaluates the efficiency of detecting local hosts infected with IP-spoofed DDoS malware.

The test environment for the assessment of the proposed algorithm was a Gigabit Ethernet network environment with IP-spoofed DDoS malware installed in the server farm of the DMZ network. For the installation of the proposed algorithm, the system configuration included an Intel i7 CPU with eight cores, 16 GB of RAM, and 2Tbyte of HDD. The Java programming language was used to program the prototype in order to verify the performance of the proposed algorithm.

Table 4 Detection of IP-spoofed DDoS attack

Algorithm 2 : Detection of IP-Spoofed DDoS Attack

Input *: DMF Table* *where DMF Table is attribute information about network traffic*
Output *: isMalwareddos, sgnVal, macVal*

```
begin
    sgnVal ← null
    macVal ← null
    isMalwareddos ← false
    r[ ][ ] ← EntryPoint(DMF Table)

    while r is not the end of r do
        i ← Ø
        if { TIN is zero and RATD is zero and ACTTI ≤ 1 } then
        //TIN is zero and RATD is zero and ACTTI is less than 1. It is DDoS attack.
            cmpMac ← r[i-1][6]
            cmpSrcIP ← r[i-1][2]
            if { cmpMac is the same with r[i][6] and  cmpSrcIP is not the same with r[i][2] } then
            //MAC address is the same, but source IP is not the same. It is IP-Spoofed
                isMalwareddos ← true
                sgnVal ← r[i][1]
                macVal ← r[i][6]
            end
        end
    i ← i + 1
    end
    return isMalwareddos, sgnVal, macVal
end
```

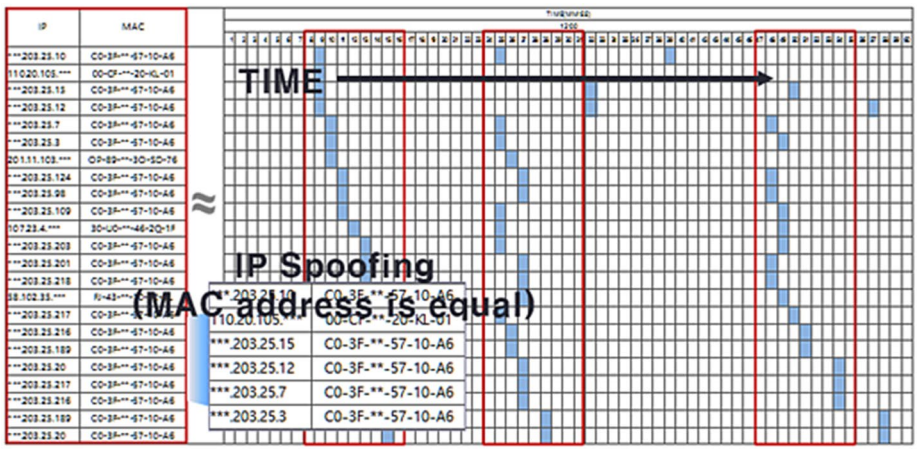

Fig. 12 Timeline of an IP-spoofed DDoS attack in a malware-infected System

Table 5 Finding of a system infected by malware

Algorithm 3 : Finding of a System Infected by Malware

```
Input : isMalwareddos, macVal          where isMalwareddos is a detection result of algorithm 2
Output : infectedHost

begin
    j ← Ø
    candidateIP[ ] ← null
    infectedHost ← null
    r[ ][ ] ← EntryPoint(DMF Table)

  if isMalwareddos is true then
    while r is not the end of r do
      i ← Ø,n ← Ø
      cndVal← r[i][10]
      if { macVal == r[i][6] and cndVal ==1 and r≠null } then
      //MAC address is the same and CND field value is 1
          candidateIP[j] ← r[i][2]          //candidateIP array stores the Source IP
          candidateDate[j++] ← r[i][8]      //candidateDate array stores the Access Time
      end
      i ← i + 1
    end

    cmpDate ← null
    foreach { candidateIP.length() ≥ n } do
      if { candidateDate[n] ≥ cmpDate } then
        cmpDate ← candidateDate[n]
        infectedHost ← candidateIP[n]      // infectedHost stores malware infection host
      end
      n ← n + 1
  return infectedHost
end
```

Comparison of the proposed method and the intrusion detection system

When malware is installed via zero-day vulnerability or drive by download, the security system can no longer guarantee the stability of the internal network. Host-based anti-malware solutions could possibly be effective, but applying it to large-scale networks poses administrative and technical difficulties. Therefore, the proposed algorithm can detect network based IP-spoofed DDoS attacks and effectively detect malware-infected hosts. Table 6 compares the performance of the proposed algorithm and the IDS.

Results of the verification of the proposed algorithm and the IDS on a testing environment showed that both were able to adequately detect DDoS attacks; however, the IDS could not detect IP-spoofed DDoS attacks and malware-infected hosts.

Figure 15a is the result of DDoS attacks detected by the IDS, and Fig. 15b shows the time that DDoS attacks were detected. Figure 15c shows the results of IP-spoofed DDoS attacks detected by prototype implemented with the proposed algorithm. Figure 15d shows the detection times of the IP-spoofed DDoS attacks and false positives in the test environment.

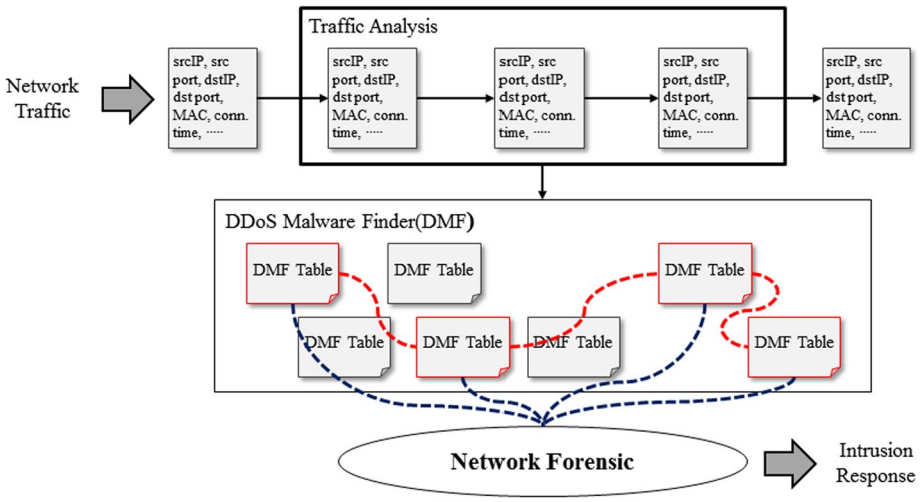

Fig. 13 Network forensic using DMF table

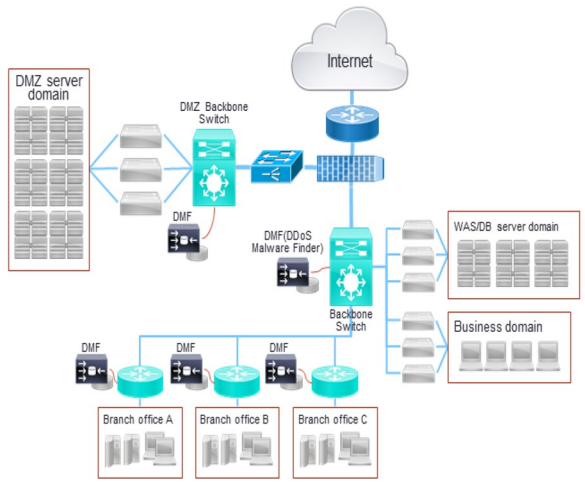

Fig. 14 Diagram of experiment environment

Detection time of IP-spoofed DDoS malware

The purpose of this study is to detect and rapidly respond to malware before network resources are depleted when massive volumes of DDoS attacks are launched by a system infected with IP-spoofed DDoS malware. Tests were performed using the proposed method, and the detection times of DDoS attacks are as shown in Fig. 16.

Figure 16 shows the detection time for IP-spoofed DDoS attacks and malware-infected systems. Graph A represents the time at which IP-spoofed DDoS attacks occur due to the local host being infected with malware, while Graph B is the discovery time of when the local host has been infected with IP-spoofed DDoS malware.

The average detection time under the proposed algorithm for IP-spoofed DDoS attacks in the local host was 98 s; the longest time was 179 s. Figure 17 shows the detection mechanism for DDoS attacks. The proposed algorithm analyzes the headers of real-time traffic to detect IP-spoofed DDoS attacks.

Table 6 The performance comparison of the proposed algorithm and IDS

Performance	Proposed algorithm	Intrusion detection system (IDS)
Detection of DDoS attack	Support	Support
Detection of IP-spoofed DDoS attack	Support	Not supported
Detection of system infected by malware	Support	Not supported

Figure 18 shows the packet information collected by the DMF. The average time taken to detect systems infected with IP-spoofed DDoS malware was 444 s; the longest time was 600 s.

Accuracy

To assess the accuracy of detecting malware-infected systems using the proposed method, tests were carried out under the test environment in Fig. 14. Accuracy tests were performed under the following conditions:

- Installation of malware in the test server and launch of IP-spoofed DDoS attacks.
- Analysis of attack features and false positives after varying the IP address of the test server.
- Analysis of attack features and false positives after replacing the server hardware.

The accuracy of detecting malware-infected systems was calculated as follows:

$$Accuracy\,(\%) = 1 - \left(\frac{false\ positive}{test\ number}\right) \times 100$$

The experimental results are presented in Table 7.

The accuracy of detecting systems infected with IP-spoofed DDoS malware was assessed under the test environment described above, and averages were obtained from 1000 runs. The detection accuracy was approximately 98% and 12 false positives occurred during the experiment. False positives were detected when there was an error with the network configuration or when ARP spoofing occurred.

Effectiveness

Past research has focused on the detection of outbound-to-inbound DDoS attacks or malware-infected hosts in internal network. However, various challenges have yet to be addressed for detecting and responding to inbound-to-outbound DDoS attacks launched by malware-infected local hosts. If DDoS attacks originating from local hosts cannot be effectively resolved, normal network services become difficult due to bandwidth exhaustion. This experiment found that the malware-infected system generated more than one million SYN flooding packets per minute and the network traffic was approximately 64 GB.

Agents do not have to be separately installed in order to detect malware-infected systems using the proposed method, and widespread detection is possible for malware-infected systems running on networks. As shown in Fig. 19, the average time taken to detect a system infected with IP-spoofed DDoS malware was 542 s.

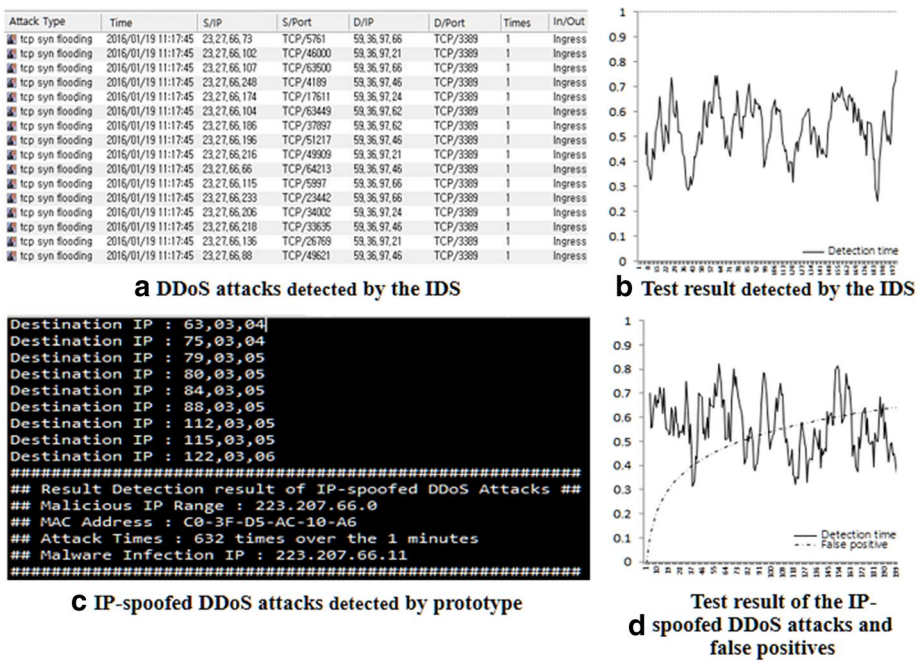

Fig. 15 A comparative analysis of the proposed algorithm and IDS

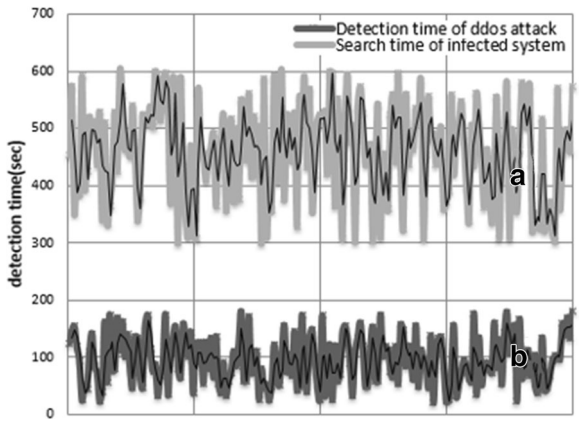

Fig. 16 Test result (detection time of infected system by malware)

$$\text{Mean time for intrusion response (MTIR)} = \text{detection time of IP spoofing DDoS attack} + \text{detection time of infected system}$$

False positives

This proposed method enables the effective detection of IP-spoofed DDoS attacks and malware-infected systems. False positives were detected in two cases. In the first case, errors in network configuration after server replacement caused anomalous traffic. In

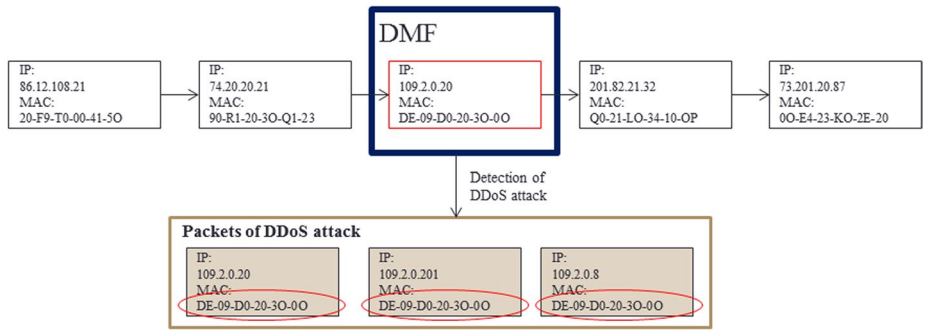

Fig. 17 Detection of IP-spoofed DDoS attacks using DMF algorithm

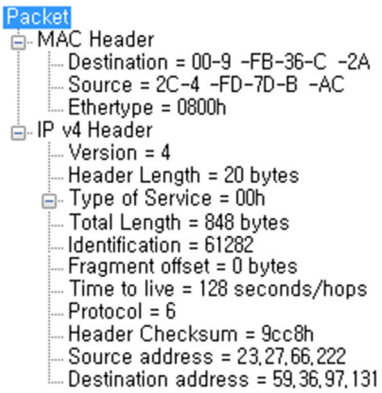

Fig. 18 Packet information collected by DMF

the second case, changes in the MAC address under ARP spoofing resulted in anomalous traffic. In actual information system environments, errors in network configuration lead to problems in communication, thereby allowing them to be identified before operating information services. The detection of ARP spoofing indicates an intrusion of the internal network and adequate measures must be taken to remove the malware.

Conclusion

A key factor in enhancing corporate reliability is the stability of information services provided through the Internet. While various security solutions are available today, DDoS attacks still pose significant threats to information service providers and security administrators.

When a system infected with IP-spoofed DDoS malware launches massive volumes of DDoS traffic in the internal network, a fundamental solution is to detect and remove the malware-infected system. This is because flooding packets generated by the local host affect network communications using the Interior Gateway Protocol (IGP) as well as the Exterior Gateway Protocol (EGP). A real-world example is the 6-h halting of network services by an Internet service provider under an IP-spoofed DDoS attack despite being equipped with various security solutions, such as DDoS countering equipment and an intrusion prevention system (Strayer et al. 2006; Stone-Gross et al. 2009). If DDoS

Table 7 Accuracy of experiment result

Error ratio	Accuracy (%)	Explanation
12:1000	98.8	TP rate = 0.988, FP = 12
8:600	98.7	TP rate = 0.987, FP = 8
2:400	99.5	TP rate = 0.995, FP = 2
1:100	99.0	TP rate = 0.99, FP = 1
0:50	100	TP rate = 1, FP = 0

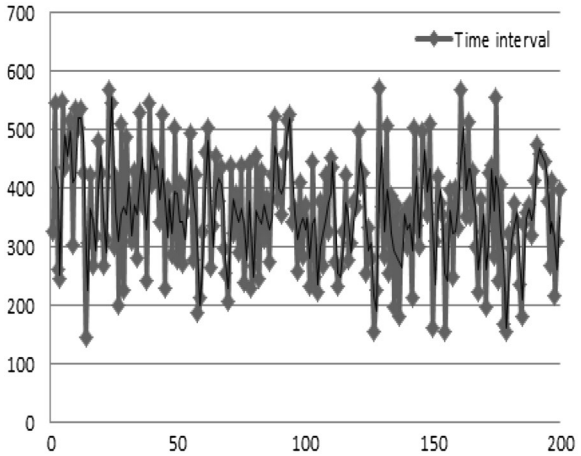

Fig. 19 Time interval to detect the system infected by malware

attacks are not detected in advance and responded to, there is likely to be a negative impact on the reliability of information services provided by such companies.

Using the algorithm proposed in this paper, it is possible to effectively detect IP-spoofed DDoS attacks and rapidly respond to malware-infected systems. The proposed algorithm derives patterns from features of IP-spoofed DDoS malware and matches information in the headers of real-time traffic with DMF table property values to detect IP-spoofed DDoS attacks and malware-infected systems. The efficiency of the proposed algorithm was demonstrated through various experiments and its applicability to actual operating environments was verified under a testbed environment.

Authors' contributions

Conceived and designed the experiments: JWS, SJL. Performed the experiments: JWS, SJL. Analyzed the data: JWS, SJL. Wrote the paper: JWS, SJL. Both authors read and approved the final manuscript.

Acknowledgements

The authors wish to thank their colleagues for insightful discussions and their feedback at various stages of the research. They also wish to thank the anonymous reviewers for their suggestions and feedback to improve the paper.

Competing interests

Both authors declare that they have no competing interests.

Received: 15 May 2016 Accepted: 18 October 2016
Published online: 26 October 2016

References

Abu Rajab M, Zarfoss J, Monrose F, Terzis A (2006) A multifaceted approach to understanding the botnet phenomenon. In: Proceedings of the 6th ACM SIGCOMM on internet measurement—IMC'06. doi:10.1145/1177080.1177086

Akamai (2015) XOR DDoS threat advisory—the Akamai blog. In: Blogs.akamai.com. https://blogs.akamai.com/2015/09/xor-ddos-threat-advisory.html. Accessed 7 Feb 2016

Arbor Networks (2015) Arbor networks detects largest ever DDoS attack in Q1 2015 DDoS report—Arbor Networks. In: Arbornetworks.com. https://www.arbornetworks.com/arbor-networks-detects-largest-ever-ddos-attack-in-q1-2015-ddos-report. Accessed 5 Feb 2016

Baker F, Savola P (2004) Ingress filtering for multihomed networks. doi:10.17487/rfc370410.17487/rfc3704

Beverly R, Berger A, Hyun Y, Claffy K (2009) Understanding the efficacy of deployed internet source address validation filtering. In: Proceedings of the 9th ACM SIGCOMM conference on internet measurement conference—IMC'09. doi:10.1145/1644893.1644936

Bremler-Barr A, Levy H (2005) Spoofing prevention method. In: Proceedings IEEE 24th annual joint conference of the IEEE computer and communications societies. doi:10.1109/infcom.2005.1497921

Duan Zhenhai, Yuan Xin, Chandrashekar J (2008) Controlling IP spoofing through interdomain packet filters. IEEE Trans Dependable Secure Comput 5(1):22–36. doi:10.1109/tdsc.2007.70224

Feinstein L, Schnackenberg D, Balupari R, Kindred D (2003) Statistical approaches to DDoS attack detection and response. In: Proceedings DARPA information survivability conference and exposition. doi:10.1109/discex.2003.1194894

Ferguson P, Senie D (2000) Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. doi:10.17487/rfc2827

François J, Wang S, State R, Engel T (2011) BotTrack: tracking botnets using NetFlow and PageRank. In: Lecture notes in computer science, pp 1–14. doi:10.1007/978-3-642-20757-0_1

Freiling FC, Holz T, Wicherski G (2005) Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks. In: Lecture notes in computer science, pp 319–335. doi:10.1007/11555827_19

Gu G, Perdisci R, Zhang J, Lee W (2008) BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Usenix.org. https://www.usenix.org/legacy/event/sec08/tech/full_papers/gu/gu_html/. Accessed 18 Feb 2016

John W, Tafvelin S (2007) Analysis of internet backbone traffic and header anomalies observed. In: Proceedings of the 7th ACM SIGCOMM conference on internet measurement—IMC'07. doi:10.1145/1298306.1298321

Liu B, Bi J (2015) DISCS: a distributed collaboration system for inter-AS spoofing defense. In: 2015 44th international conference on parallel processing. doi:10.1109/icpp.2015.25

Park K, Lee H (2001) On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. SIGCOMM Comput Commun Rev 31(4):15–26. doi:10.1145/964723.383061

Perdisci R, Ariu D, Giacinto G (2013) Scalable fine-grained behavioral clustering of HTTP-based malware. Comput Netw 57(2):487–500. doi:10.1016/j.comnet.2012.06.022

Sakib MN, Huang C-T (2016) Using anomaly detection based techniques to detect HTTP-based botnet C&C traffic. In: 2016 IEEE international conference on communications (ICC). doi:10.1109/icc.2016.7510883

Stone-Gross B, Cova M, Cavallaro L, Gilbert B, Szydlowski M, Kemmerer R, Vigna G (2009) Your botnet is my botnet. In: Proceedings of the 16th ACM conference on computer and communications security—CCS'09. doi:10.1145/1653662.1653738

Strayer W, Walsh R, Livadas C, Lapsley D (2006) Detecting botnets with tight command and control. In: Proceedings. 2006 31st IEEE conference on local computer networks. doi:10.1109/lcn.2006.322100

Tegeler F, Fu X, Vigna G, Kruegel C (2012) BotFinder. In: Proceedings of the 8th international conference on emerging networking experiments and technologies—CoNEXT'12. doi:10.1145/2413176.2413217

Wang H, Jin C, Shin KG (2007) Defense against spoofed IP traffic using hop-count filtering. IEEE/ACM Trans Netw 15(1):40–53. doi:10.1109/tnet.2006.890133

Wang B, Li Z, Tu H, Ma J (2009) Measuring Peer-to-Peer Botnets Using Control Flow Stability. In: 2009 International Conference on Availability, Reliability and Security. doi:10.1109/ares.2009.59

Zeidanloo HR, Bt Manaf A, Vahdani P, Tabatabaei F, Zamani M (2010) Botnet detection based on traffic monitoring. In: 2010 International Conference on Networking and Information Technology. doi:10.1109/icnit.2010.5508552

Zhao D, Traore I, Sayed B, Lu W, Saad S, Ghorbani A, Garant D (2013) Botnet detection based on traffic behavior analysis and flow intervals. Comput Secur 39:2–16. doi:10.1016/j.cose.2013.04.007