IPSJ Journal Vol. 52 No. 4 1706–1716 (Apr. 2011)

Regular Paper

Detection of Bot Infected PC Using Destination-based IP Address and Domain Name Whitelists

Keisuke Takemori,[†1] Takahiro Sakai,[†2]
Masakatsu Nishigaki[†2] and Yutaka Miyake[†1]

As a bot communicates with a malicious controller over a normal communication or an encrypted channel and updates its code frequently, it becomes difficult to detect an infected personal computer (PC) using a signature-based intrusion detection system (IDS) and an antivirus system (AV). As sending control and attack packets from the bot process are independent of the user operation, a behavior monitor is effective in detecting an anomaly communication. In this paper, we propose a bot detection technique that checks outbound packets with destination-based whitelists. If any outbound packets during the non-operating duration do not match the whitelists, the PC is considered to be infected by the bot. The whitelists are a set of legitimate IP addresses (IPs) and/or domain names (DNs). We implement the proposal system as a host-based detector and evaluate false negative (FN) and false positive (FP) performance.

1. Introduction

Spam e-mails and DDoS attacks have now become critical issues to the Internet. These attacks are considered to be sent from bot infected PCs. Some studies[1)–9)] analyze bot activities and report that many PCs are infected with the bot and controlled via command and control (C&C) servers to distribute attack packets. Because the bot communicates with the C&C server that is controlled by a herder over a normal communication or an encrypted channel, it is hard to detect the control channel via signature-based IDS. As the number of unknown bots is programmed and their codes are updated frequently, it becomes difficult to detect infected PCs via a signature-based AV. Since the user takes no countermeasures for the infected PC, this then constitutes a botnet.

As sending control and attack packets which are sent by the bot process are independent of the user operation, a behavior monitor is effective to detect anomaly activities. An anomaly detection technique that uses a process-based whitelist has been proposed[10)]. If an unknown process is invoked, the PC is considered to be infected by a bot. A file-based whitelist is used in Ref. 11), and any anomaly changes in system files are detected. These approaches are classified as anomaly extrusion detection on the infected PC. However, many kinds of applications and plug-in functions are installed on a PC, so it is difficult to distinguish a bot from legitimated software.

In other words, "how to make a good whitelist" is one of the key factors in anomaly detection techniques. Since conventional anomaly detection techniques have used simple and static whitelists, in this paper, we propose a bot detection technique with sophisticated and dynamic destination-based whitelists. Our whitelists are "sophisticated" because we use multiple whitelists, activated only during the non-operating duration of the PC, to minimize the FN and FP. To be more precise, as daemon processes access well-known sites cyclically, as the PC communicates with servers on the LAN such as DNS servers automatically, and as user applications communicate with servers on the Internet manually, IPs and/or DNs of the daemon access sites, LAN+DNS servers, and user access servers are registered in the daemon access, LAN+DNS, and user access whitelists, respectively. Our whitelists are "dynamic" because the user access and LAN+DNS whitelists can be updated automatically by monitoring user operations. The technique contributes not only to enhancing the detection accuracy but also to minimizing the maintenance costs of the whitelists. Our technique checks the addresses of outbound packets from the PC with destination-based whitelists. If any addresses of outbound packets during the non-operating duration do not match the whitelists, the PC is considered to be infected by a bot. Another advantage of our whitelists is that when many IPs and DNs in the whitelists are grouped into a few sub-networks, we can describe the whitelists compactly with start to end IPs and/or superior DNs. Our technique is implemented on a Windows XP PC as an anomaly extrusion detector. We investigate the sizes of

†1 KDDI R&D Laboratories Inc.
†2 Graduate School of Science and Technology, Shizuoka University

the whitelists to consider how easy it is to maintain. Also, we evaluate the FN for detection of bot activities and the FP under user operation.

The rest of this paper is organized as follows. In Section 2, we investigate the bot communication patterns using the honeypot. Section 3 surveys conventional tools and considers their problems. In Section 4, we propose and implement the bot detection system using the destination-based whitelists. Section 5 shows evaluation results of FN and FP counts. Section 6 considers the limitations of our approach and Section 7 concludes the paper.

2. Investigation of Bot Communication Activities

We have collected 44 kinds of initial bot codes by using the Nepenthes honeypot [12] and analyzed bot activities by using a virtual machine [13].

Figures 1 and **2** show an example of a bot communication pattern that was monitored on our honeypot system. A victim PC was intruded and an initial bot code "˜21.tmp.exe" was installed by an attacker. The "˜21.tmp.exe" accessed three C&C and download servers to download both a control policy and a new bot code "nbin.exe". After 53 seconds, the "nbin.exe" was invoked and accessed a download server in order to update a new bot code "EventLogger.exe". After 90 seconds, the "EventLogger.exe" was invoked and distributed attack packets to other PCs. Here, the attack packets were dropped by our honeypot system.

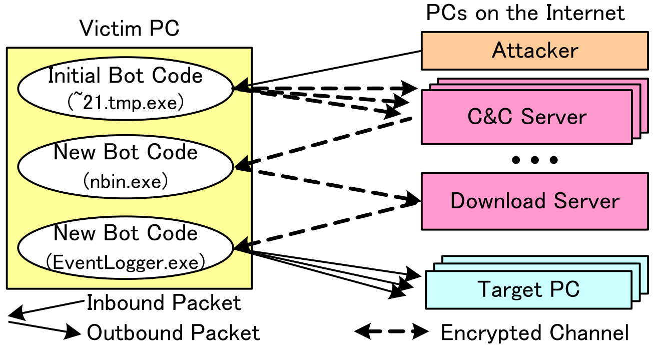
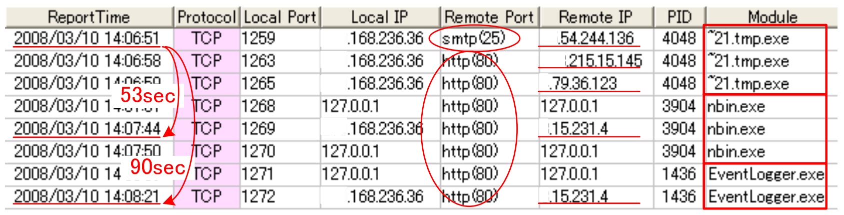


Fig. 1 Example of bot communication pattern.



ReportTime	Protocol	Local Port	Local IP	Remote Port	Remote IP	PID	Module
2008/03/10 14:06:51	TCP	1259	.168.236.36	smtp(25)	.54.244.136	4048	˜21.tmp.exe
2008/03/10 14:06:58	TCP	1263	.168.236.36	http(80)	.215.15.145	4048	˜21.tmp.exe
2008/03/10 14:06:59	TCP	1265	.168.236.36	http(80)	.79.36.123	4048	˜21.tmp.exe
2008/03/10 1	TCP	1268	127.0.0.1	http(80)	127.0.0.1	3904	nbin.exe
2008/03/10 14:07:44	TCP	1269	.168.236.36	http(80)	.15.231.4	3904	nbin.exe
2008/03/10 14:07:50	TCP	1270	127.0.0.1	http(80)	127.0.0.1	3904	nbin.exe
2008/03/10 14	TCP	1271	127.0.0.1	http(80)	127.0.0.1	1436	EventLogger.exe
2008/03/10 14:08:21	TCP	1272	.168.236.36	http(80)	.15.231.4	1436	EventLogger.exe

Fig. 2 Bot communication monitor on our honeypot.

The 44 binary files include known bots, unknown bots, and incomplete codes. 37 of the binary files could be invoked on the virtual machine and communicated with the Internet servers to download the new bot codes. **Table 1** shows the intervals between the first outbound packets from the initial bot code and the packets from the new code. Also, counts of bot access unknown-IPs for 10 minutes after the click of the binary files are shown. All of the 37 bots communicated with many C&C servers. Mean, minimum, and maximum access unknown-IPs are 60, 4, and 162 respectively.

3. Conventional Tools

In this section, we review conventional IDS and AV tools and suggest the difficulties of detecting bot infected PCs.

3.1 IDS

Most IDSs can detect known attack packets using a signature database. In general, the infected PC is controlled over an Internet relay chat (IRC) channel, thus bot IRC detectors are proposed [7),8)]. In addition, various IDS events are correlated to detect bot communication patterns [9)]. Reference 14) has a malware virtual machine filter that can achieve a deep packet inspection by executing potentially malicious binaries or web pages. Several of the latest IDSes can monitor encrypted communication channels using pre-assigned decryption keys.

Table 1 Bot communication activities.

	Mean	Minimum	Maximum
Communication Interval between Previous and New Bot-codes	63 sec	17 sec	130 sec
Count of Access Unknown-IPs	60	4	162

As their monitoring techniques can be implemented as a network gateway using a high performance CPU, it is not appropriate for a host-based infected detection.

If an infected PC communicates with C&C servers over a non-IRC channel or an encrypted channel, and if the infected PC sends spam e-mails or DDoS attacks with normal protocol, it is hard to detect the control and/or attack packets using the signature-based IDS.

3.2 AV

Most AVs can detect the known bot code using a virus signature and have the advantage of detecting the known bot code with few FP.

As a herder programs and updates unknown bot codes frequently, it is hard to release new signatures immediately. The number of registered signatures should be limited to achieve quick matching. As a herder releases numerous kinds of bot code, a minor bot code may not be included on the virus signature. Furthermore, many bot codes manipulate the "hosts" file to inhibit updating the signatures of AV, e.g., "127.0.0.1 signature-download-site" [15].

3.3 Extrusion Detection on PC

An extrusion detection technique that uses a whitelist of PC processes has been proposed [10]. If an unknown process is invoked, the PC is considered to be infected by a bot. The technique can detect bot infections when the process-based whitelist is properly maintained.

However, it is difficult to distinguish a bot code from legitimate software because many kinds of applications are installed on the PC. If a plug-in function is installed on a web browser or a messenger service, it is hard to recognize this as a legitimate process. Furthermore, as there are many trojan-type bots that infect legitimate applications, e.g., *explorer.exe*, *cmd.exe*, and *notepad.exe*, it is also hard to distinguish between normal and infected codes [15].

4. Proposal and Implementation

Because the infected PC communicates with C&C servers and distributes attack packets to many hosts, we propose an extrusion detection technique that monitors outbound packets from the PC to an unknown destination IP and/or DN.

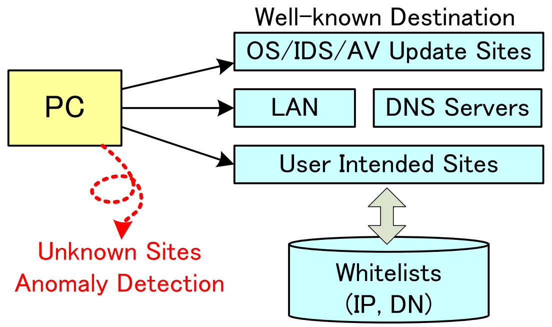


Fig. 3 Destination-based anomaly detection.

4.1 Design of Anomaly Detection

Figure 3 shows our proposed model, which manages whitelists to detect anomalous PC communications. The PC accesses legitimate servers that distribute patch files of the OS and the signatures of IDS/AV automatically. When an un-infected PC accesses Internet servers, the IP is retrieved from a DNS server that is operated by an ISP or a local area network (LAN). Also, packets of ARP, 135/TCP, and other kinds of protocol packets are frequently exchanged on the LAN. Consequently, the whitelists manage well-known destination IPs and/or DNs that include OS/IDS/AV upload sites, DNS servers, and LAN addresses. When a user operates the PC, e.g., web browsing and sending e-mails, the outbound packets are sent to various servers on the Internet, meaning an outbound packet monitor needs to exclude outbound packets that are sent by the user operation.

It is noted that in our technique, each PC generates its own whitelists respectively, which are not shared by several PCs. In other words, our technique is classified as a host-based infected detection, in which anomaly packets are detected only on each PC.

Figure 4 shows our anomaly detection technique. A key operation monitor checks the user operation and starts a timer that defines the non-operating duration. We implement the key operation monitor on the Windows XP using the "SetWindowHookEx()" API to hook the keyboard operations. An outbound packet monitor compares the whitelists with outbound packets that are sent dur-

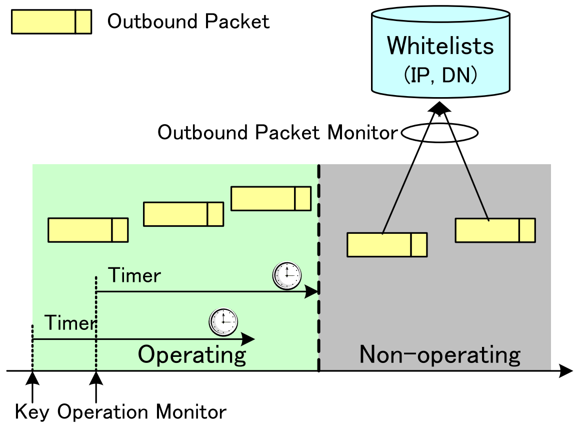


Fig. 4 Anomaly detection during non-operating duration.

ing the non-operating duration. The outbound packet monitor is implemented using the "packet.dll" to hook the packets.

If the destination IPs and/or DNs of the outbound packets during non-operating duration are not listed on the whitelists, the PC is considered to be infected by a bot.

4.2 Design of Destination-based Whitelists

Here, we consider how to make the whitelists. Firstly, we classified the whitelists into groups of daemon access sites, LAN+DNS, and user access sites. The daemon access whitelist manages the well-known Internet domains, and the LAN+DNS whitelist manages the LAN and DNS servers. Also, the user access whitelist manages the IPs of outbound packets by the user operation.

4.2.1 Daemon Access Whitelist

We assume that new PCs are not infected by a malicious code. The automatic outbound packets from the new PC that includes OS and legitimate applications are only sent to well-known legitimate domains. Thus, we leave the new PC for a few days under non-operating circumstances, and capture the outbound packets in order to list destination IPs and/or DNs. We then register these legitimate domains in the daemon access whitelist manually.

In addition, when a user installs new applications that communicate automatically with outside hosts, the legitimacy of the hosts is investigated using a reputation site, e.g., MalwareDomainList [16]. If the hosts are not malicious sites, the IPs and/or DNs of the hosts are listed in the daemon access whitelist.

4.2.2 LAN+DNS Whitelist

In the case of a mobile PC, certain LAN and DNS server addresses are assigned by a DHCP server because a user carries the mobile PC to various locations. Thus, we implement a configuration monitor of a network interface card that is capable of dumping the LAN and the DNS server addresses by using "GetNetworkParams()" and "WSAIoctl()" APIs. The LAN and DNS server addresses are automatically registered in the LAN+DNS whitelist.

4.2.3 User Access Whitelist

In the case of an Asynchronous JavaScript+XML (Ajax), a web browser downloads an Ajax module that communicates with only an Ajax download site asynchronously. In the case of a chat application, a chat client connects to a chat server and exchanges packets automatically. These applications were invoked by the user and communicate with fixed sites. Thus, the IPs of outbound packets during the operating duration are registered in a user access whitelist automatically. Usually Ajax/chat functions are closed while users stop operations. After sufficient time has elapsed, e.g., 24 hours, the IPs and/or FQDNs can be removed automatically in order to minimize the size of the user access whitelist..

4.3 Superior DN and IP Range

After a few days under non-operating circumstances, many kinds of IPs are listed in the daemon whitelist, because there are many cache servers on the Internet[17]. **Figure 5** shows procedures used to list up the daemon access, the LAN+DNS, and the user access whitelists. Most IPs are linked with some DNs, and the DNs are grouped into a few kinds of superior DNs using the wild card "*" that masks the host names.

In general, OS, IDS and AV automatically download patch and signatures via cache services. In the case of the Akamai cache service [17], 14,000 servers on 1,100 domains applied for the download servers. The patch and/or signature downloader accesses download servers that include real and cache servers indicated by FQDN. The total number of FQDN is in less than hundreds, while the number of DN that is indicated by "*" will be fewer than 10.

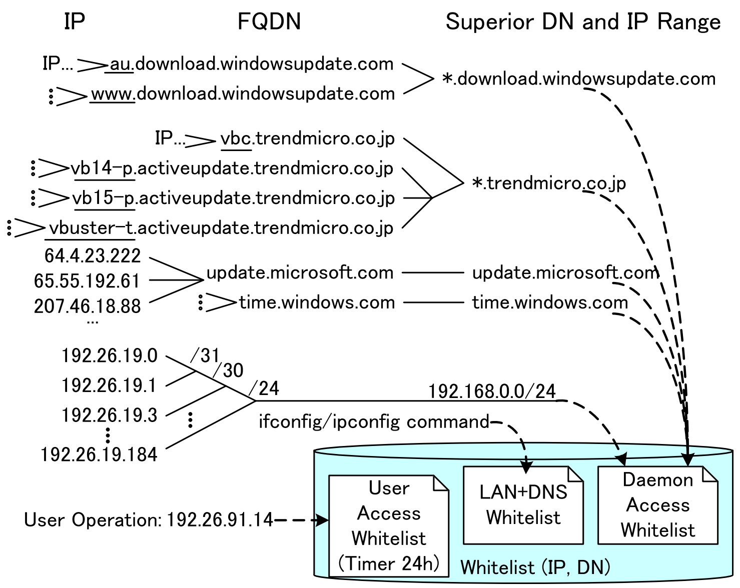


Fig. 5 Procedures to list up whitelists.

The IPs listed in the user and daemon access whitelists can be summarized using a subnet mask. When neighboring IPs are grouped by decreasing value of the subnet mask, the IP range will be increased. Also, the FQDNs listed in the whitelists can be summarized by erasing a host name or sub domain name. The summarization flow is shown in **Fig. 6**. Here, IPs of the LAN+DNS whitelist can be defined by using a control command for a network interface card.

4.4 Registration Interface

In **Fig. 7**, we implemented a registration interface to maintain the daemon access whitelist that is only maintained manually. The interface registers the keywords of a protocol, destination port, destination IP range, destination DN, and process information. These keywords are set as the "AND" condition to match outbound packets.

4.5 IP to DN Translator

Because the proposed technique retrieves the IP of the outbound packet from the daemon access whitelist, a translator from the IP to the DN should be implemented.

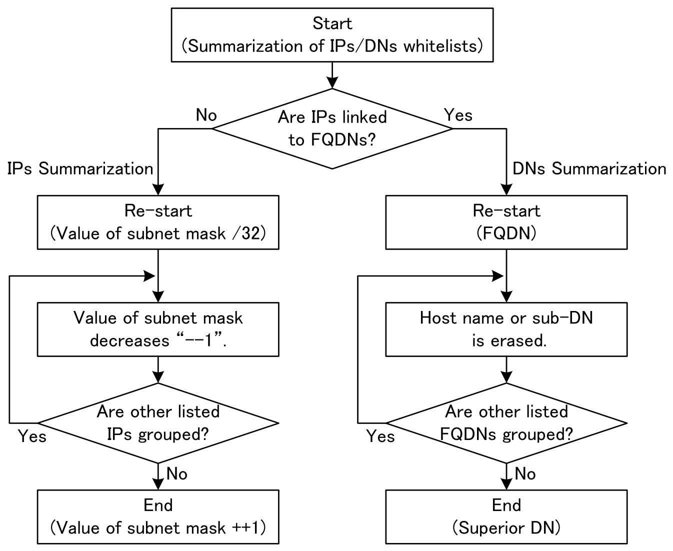
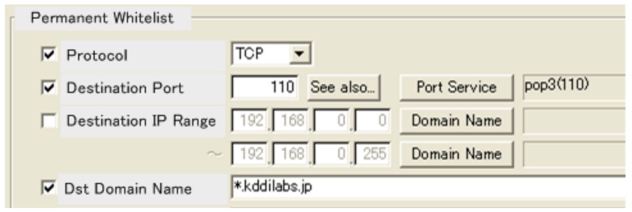

Start
(Summarization of IPs/DNs whitelists)
Are IPs linked to FQDNs?
No
Yes
IPs Summarization
DNs Summarization
Re-start
(Value of subnet mask /32)
Re-start
(FQDN)
Value of subnet mask decreases "−−1".
Host name or sub-DN is erased.
Are other listed IPs grouped?
Are other listed FQDNs grouped?
Yes
No
Yes
No
End
(Value of subnet mask ++1)
End
(Superior DN)

Fig. 6 Summarization flow of IPs and DNs.


Permanent Whitelist

☑ Protocol	TCP			
☑ Destination Port	110	See also...	Port Service	pop3(110)
☐ Destination IP Range	192 . 168 . 0 . 0		Domain Name	
~	192 . 168 . 0 . 255		Domain Name	
☑ Dst Domain Name	*.kddilabs.jp			

Fig. 7 Registration interface for daemon access whitelist.

Figure 8 shows the translation procedure from the IP to the DN. At the start of communication, most PCs send the query packet to the DNS server to retrieve the IP from the FQDN. Our technique monitors a DNS response packet that includes the IPs and the FQDN and a combination of the IPs and the FQDN is stored in an IP-to-DN translator. After receiving the DNS response packet, the PC communicates with a destination host using the IP. Our system can match the IPs of outbound packets with the DNs in the daemon access whitelist by

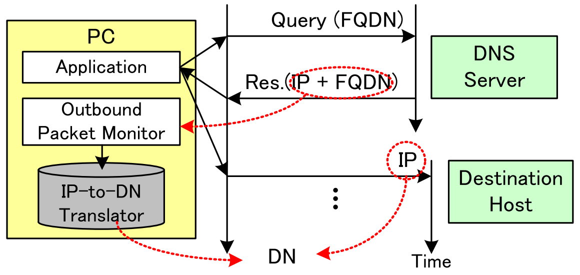


Fig. 8 IP-to-DN translation procedure.

using the IP-to-DN translator.

Because one FQDN is assigned to many IPs, multiple IP addresses are replied in the DNS query response. In addition, there is sometimes a time-lag between the DNS response and destination access caused by DNS prefetching for several prospective web sites. The IP-to-DN translator records and retains all retrieved IPs to achieve the translation as far as possible. The IPs and DNs retained in the IP-to-DN translator, including the redundancy information, should be erased after a sufficient period, e.g., 24 hours has elapsed.

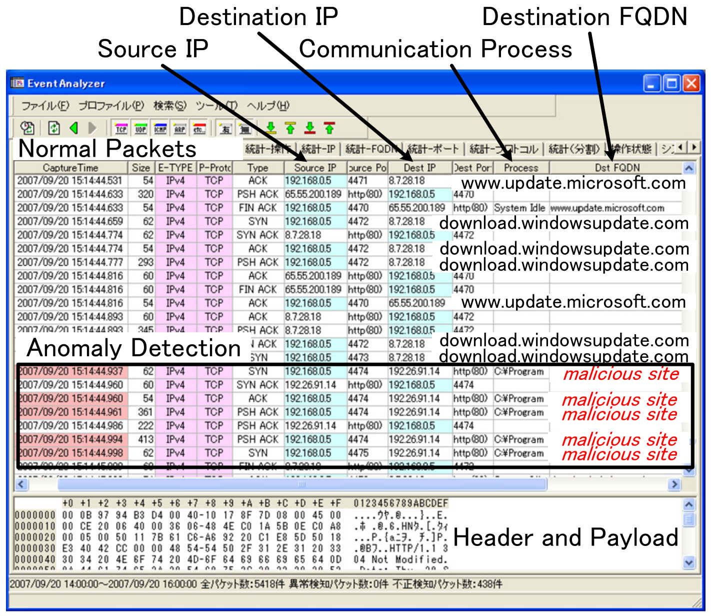
4.6 Communication Process Monitor

It is useful to suggest a communication process that sends packets to the unknown IP or DN. A PC user refers to a communication process in order to distinguish un-infected and infected status. Also, it is effective in suspending the bot process for a swift response. We implement a communication process monitor on the Windows XP, which calls the following APIs in order to hook communicating process names.

- TCP: AllocateAndGetTcpExTableFromStack().
- UDP: AllocateAndGetUdpExTableFromStack().

4.7 Alarm Interfaces

Figure 9 shows an alarm interface that indicates the source IP, the destination IP, the destination FQDN, and the communication process to notify related information. The figure indicates anomaly packets depicted by a red color. The “www.update.microsoft.com” and “download.windows.update” are presented as normal packets because both sites are listed in the daemon access whitelist. On the other hand, the outbound packets to the “malicious” during non-operating duration are indicated as an anomaly, because the site is not listed in the whitelists.


Destination IP
Destination FQDN
Source IP
Communication Process
Event Analyzer
ファイル(F) プロファイル(P) 検索(S) ツール(T) ヘルプ(H)
Normal Packets
統計-操作 | 統計-IP | 統計-FQDN | 統計-ポート | 統計-プロトコル | 統計(分割) | 操作状態
Anomaly Detection
www.update.microsoft.com
download.windowsupdate.com
download.windowsupdate.com
download.windowsupdate.com
www.update.microsoft.com
download.windowsupdate.com
download.windowsupdate.com
malicious site
malicious site
malicious site
malicious site
malicious site
Header and Payload

CaptureTime	Size	E-TYPE	P-Proto	Type	Source IP	ource Po	Dest IP	Dest Por	Process	Dst FQDN
2007/09/20 15:14:44.531	54	IPv4	TCP	ACK	192.168.0.5	4471	8.7.28.18			
2007/09/20 15:14:44.633	320	IPv4	TCP	PSH ACK	65.55.200.189	http(80)	192.168.0.5	4470		
2007/09/20 15:14:44.633	54	IPv4	TCP	FIN ACK	192.168.0.5	4470	65.55.200.189	http(80)	System Idle	www.update.microsoft.com
2007/09/20 15:14:44.659	62	IPv4	TCP	SYN	192.168.0.5	4472	8.7.28.18			
2007/09/20 15:14:44.774	62	IPv4	TCP	SYN ACK	8.7.28.18	http(80)	192.168.0.5	4472		
2007/09/20 15:14:44.774	54	IPv4	TCP	ACK	192.168.0.5	4472	8.7.28.18			
2007/09/20 15:14:44.777	293	IPv4	TCP	PSH ACK	192.168.0.5	4472	8.7.28.18			
2007/09/20 15:14:44.816	60	IPv4	TCP	ACK	65.55.200.189	http(80)	192.168.0.5	4470		
2007/09/20 15:14:44.816	60	IPv4	TCP	FIN ACK	65.55.200.189	http(80)	192.168.0.5	4470		
2007/09/20 15:14:44.816	54	IPv4	TCP	ACK	192.168.0.5	4470	65.55.200.189			
2007/09/20 15:14:44.893	60	IPv4	TCP	ACK	8.7.28.18	http(80)	192.168.0.5	4472		
2007/09/20 15:14:44.893	345	IPv4	TCP	PSH ACK	8.7.28.18	http(80)	192.168.0.5	4472		
				ACK	192.168.0.5	4472	8.7.28.18			
				SYN	192.168.0.5	4473	8.7.28.18			
2007/09/20 15:14:44.937	62	IPv4	TCP	SYN	192.168.0.5	4474	192.26.91.14	http(80)	C:¥Program	
2007/09/20 15:14:44.960	60	IPv4	TCP	SYN ACK	192.26.91.14	http(80)	192.168.0.5	4474		
2007/09/20 15:14:44.960	54	IPv4	TCP	ACK	192.168.0.5	4474	192.26.91.14	http(80)	C:¥Program	
2007/09/20 15:14:44.961	361	IPv4	TCP	PSH ACK	192.168.0.5	4474	192.26.91.14	http(80)	C:¥Program	
2007/09/20 15:14:44.986	222	IPv4	TCP	PSH ACK	192.26.91.14	http(80)	192.168.0.5	4474		
2007/09/20 15:14:44.994	413	IPv4	TCP	PSH ACK	192.168.0.5	4474	192.26.91.14	http(80)	C:¥Program	
2007/09/20 15:14:44.998	62	IPv4	TCP	SYN	192.168.0.5	4475	192.26.91.14	http(80)	C:¥Program	

```
         +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F   0123456789ABCDEF
0000000  00 0B 97 94 B3 D4 00 40-10 17 8F 7D 08 00 45 00   ....ｳﾔ.@...}..E.
0000010  00 CE 20 06 40 00 36 06-48 4E C0 1A 5B 0E C0 A8   .ﾎ .@.6.HNﾀ.[.ﾀｨ
0000020  00 05 00 50 11 7B 61 C6-A6 92 20 C1 E8 5D 50 18   ...P.{aﾆｦ. ﾁ.]P.
0000030  E3 40 42 CC 00 00 48 54-54 50 2F 31 2E 31 20 33   .@Bﾌ..HTTP/1.1 3
0000040  30 34 20 4E 6F 74 20 4D-6F 64 69 66 69 65 64 0D   04 Not Modified.
```

2007/09/20 14:00:00～2007/09/20 16:00:00 全パケット数:5418件 異常検知パケット数:0件 不正検知パケット数:438件

Fig. 9 Alarm interface.

5. Evaluations and Considerations

In this section, we consider the maintenance cost of the daemon access whitelist and evaluate FP and FN counts in order to estimate the ability of the detection of bot infection.

5.1 Condition of Evaluation

Because the minimum communication interval is 17 seconds referring Table 1, the timer that defines the non-operating duration was set to 10 seconds to reduce the FN. We evaluated the PCs of three examinees who worked in laboratories. Profiles of their Internet services are shown in **Table 2**.

Table 2 Profiles of Internet access.

	Daemon Application	User Access Application
User A	OS, AV, MS–App., PDF	Mail, Web
User B	OS, AV, MS–App., PDF, NTP	Mail, Web, VPN
User C	OS, AV, MS–App., PDF, NTP	Mail, Web, Chat

5.2 Consideration of Daemon Access Whitelist

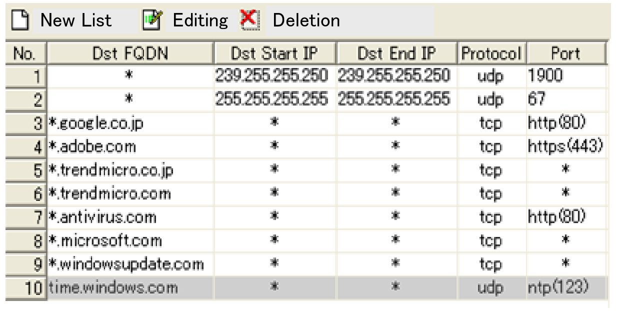
We estimate the size of the daemon access whitelist on a PC to consider how easily maintained it is. The daemon access whitelist is described with the IP range using start-IP to end-IP and/or with DN using wild card "*".

Figures 10 (a), (b), and (c) show screenshots of the daemon access whitelist of the users-A, -B, and -C PCs. These superior DNs were detected by the outbound packet monitor during a non-operating situation, when all applications without automatic start-up daemons were suspended for 5 days. In the case of user-A, a universal plug and play (UPnP) service accessed to 239.255.255.250, the PC broadcasted packets to 255.255.255.255, as well as web plug-in, PDF, AV, OS and Time applications accessed to 8 domains automatically. In the case of user-B, a mail and VPN servers were located on 192.26.91.0/24 network, as well as the web plug-in services accessed to *.mozilla.org, *.mozilla.com, and *.google.co.jp. In the case of user-C, a chat application accessed to *.yahoo.co.jp and *.yimg.jp. Here the *.yimg.jp is a part of the yahoo domain. The AV accessed to a1763.g.akamai.net and *.sourcenext.info to download the signatures. Consequently, only 10, 11, and 11 domains are listed in the daemon access whitelist on the PCs. It is easy to make and maintain the daemon access whitelist. Here, the LAN+DNS and the user access whitelists are maintained automatically.

5.3 False Negatives (FN)

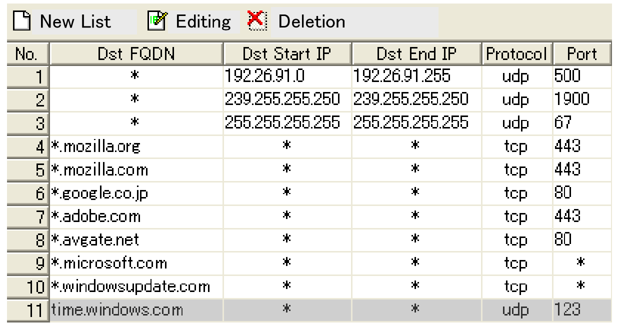
The FN of the proposed model occurs when a bot does not work on an infected PC, when a bot accesses sites that are listed on the whitelists under the non-operating duration, and when a bot accesses any sites under operating duration only.

We used all 44 kinds of binary files collected in Section 2 to evaluate the FN ratio because it is difficult to distinguish between the unknown bots and the incomplete codes.



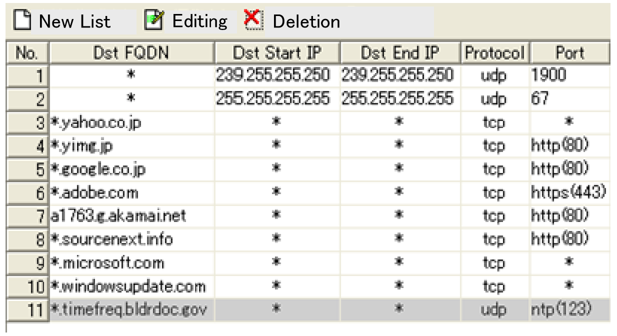
No.	Dst FQDN	Dst Start IP	Dst End IP	Protocol	Port
1	*	239.255.255.250	239.255.255.250	udp	1900
2	*	255.255.255.255	255.255.255.255	udp	67
3	*.google.co.jp	*	*	tcp	http(80)
4	*.adobe.com	*	*	tcp	https(443)
5	*.trendmicro.co.jp	*	*	tcp	*
6	*.trendmicro.com	*	*	tcp	*
7	*.antivirus.com	*	*	tcp	http(80)
8	*.microsoft.com	*	*	tcp	*
9	*.windowsupdate.com	*	*	tcp	*
10	time.windows.com	*	*	udp	ntp(123)

(a) Daemon access whitelist of user-A.



No.	Dst FQDN	Dst Start IP	Dst End IP	Protocol	Port
1	*	192.26.91.0	192.26.91.255	udp	500
2	*	239.255.255.250	239.255.255.250	udp	1900
3	*	255.255.255.255	255.255.255.255	udp	67
4	*.mozilla.org	*	*	tcp	443
5	*.mozilla.com	*	*	tcp	443
6	*.google.co.jp	*	*	tcp	80
7	*.adobe.com	*	*	tcp	443
8	*.avgate.net	*	*	tcp	80
9	*.microsoft.com	*	*	tcp	*
10	*.windowsupdate.com	*	*	tcp	*
11	time.windows.com	*	*	udp	123

(b) Daemon access whitelist of user-B.



No.	Dst FQDN	Dst Start IP	Dst End IP	Protocol	Port
1	*	239.255.255.250	239.255.255.250	udp	1900
2	*	255.255.255.255	255.255.255.255	udp	67
3	*.yahoo.co.jp	*	*	tcp	*
4	*.yimg.jp	*	*	tcp	http(80)
5	*.google.co.jp	*	*	tcp	http(80)
6	*.adobe.com	*	*	tcp	https(443)
7	a1763.g.akamai.net	*	*	tcp	http(80)
8	*.sourcenext.info	*	*	tcp	http(80)
9	*.microsoft.com	*	*	tcp	*
10	*.windowsupdate.com	*	*	tcp	*
11	*.timefreq.bldrdoc.gov	*	*	udp	ntp(123)

(c) Daemon access whitelist of user-C.

Fig. 10 Daemon access whitelist.

Table 3 FN of proposal and conventional IDSs.

	Proposal	IDS [18]	IDS [19]
Detected	37/44	26/44	20/44
FNR	16 %	41 %	55 %

Table 4 FN pattern of proposal and conventional IDSs.

Proposal	IDS [18]	IDS [19]	Counts (44)
X	X	X	20
X	X	−	6
X	−	X	0
−	X	X	0
X	−	−	11
−	−	X	0
−	X	−	0
−	−	−	7

X: Detected, −: Non-detected

Table 5 FN of proposal and conventional AVs.

	Proposal	AV [20]	AV [21]
Detected	37/44	38/44	40/44
FNR	16 %	14 %	9 %

Table 6 FN pattern of proposal and conventional AVs.

Proposal	AV [20]	AV [21]	Counts (44)
X	X	X	35
X	X	−	0
X	−	X	2
−	X	X	3
X	−	−	0
−	−	X	0
−	X	−	0
−	−	−	4

X: Detected, −: Non-detected

Tables 3 and **4** show the FN results of the proposal technique and conventional IDSs [18),19)] that monitored outbound packets from the infected virtual machine. 7 codes did not work on the virtual machine, and 37 codes accessed the complementary set of the whitelists under the non-operating duration. In particular, 11 bots could be detected by only our proposal technique. The FNR of the proposed model was 16%, while the FNRs of the conventional IDSs were larger than that of the proposal technique. It was considered that some bots accessed other PCs with normal protocol, although conventional IDSs could not detect bot activities such as spam E-mail and bot download accesses. It suggests that our technique using destination-based whitelists is effective in detecting the unknown bot activities.

Tables 5 and **6** show the FN results of the proposal technique and the conventional AVs [20),21)]. Several of the latest AVs implement both a signature-based analyzer and an anomaly memory access monitor. The anomaly-based approach has the advantage of detecting unknown codes without virus signatures. As the destination whitelists proposed in this paper are classified as a signature-based approach, at this point, we compare our technique with only the signature-based AVs in Tables 5 and 6. The FNR of the proposed model was 16%, while the FNRs of the conventional AVs were smaller than that of the proposal technique. 4 binary files were considered to be incomplete and 3 binary files did not work on a virtual machine. On the other hand, 2 binary files were non-detectable by conventional AVs [20)], but detected by the proposal technique. This suggests that our technique using the destination-based whitelists is effective in detecting the unknown bot code.

5.4 False Positives (FP)

First, we investigated how many packets were exchanged between a PC and network system automatically. We arranged three user PCs that were used for their work and monitored all inbound and outbound packets for 5 days. **Figure 11** shows all packets, TCP, UDP, ICMP, and ARP packets/day under non-operating circumstances. All applications without automatic start-up services were suspended. When users-A, -B and -C PCs were not used for a day, around 3,500, 14,900 and 7,700 packets/day were automatically exchanged. The windows updates were invoked cyclically and AVs were installed. In addition, the configurations of the network interface cards of users-B and -C PCs were set with “Microsoft network client”, “Microsoft file and printer sharing”, and “QoS packet

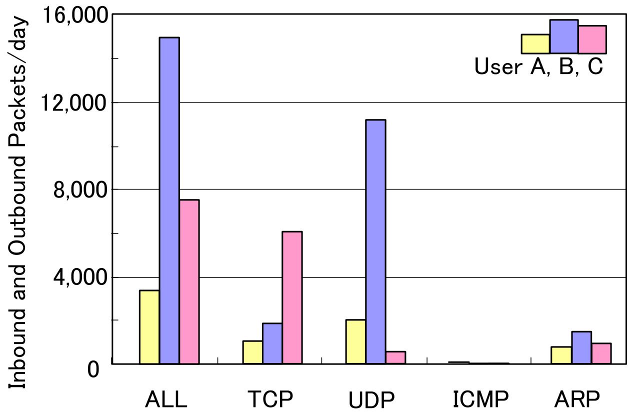


Fig. 11 All packets/day under non-operating circumstances.

Table 7 Matching counts with whitelists and FP count.

Matching Count/Day	User A	User B	User C
Daemon Access Whitelist	5.4	12.6	21.8
LAN+DNS Whitelist	5.4	21.4	55.4
User Access Whitelist	2.0	38.6	4.8
ALL Whitelists	12.8	72.6	82.4
FP Count/Day	0.0	0.0	0.4

scheduler".

Next, we evaluated the FP counts using the whitelists shown in Fig. 5. The FP of our technique occurs when the outbound packets after the timer are sent to a complementary set of the whitelists. Here, the FP count means kinds of unknown IPs, but it does not mean packet count bound for the unknown IPs. We monitored all inbound and outbound packets on users-A, -B and -C PCs for 5 working days. **Table 7** shows the matching IP count/day with the whitelists after 10 seconds and the FP count/day of the three examinees. In the case of user-A, 5.4, 5.4, and 2.0 IPs were matched with the daemon access, the LAN+DNS, and the user access whitelists respectively, and no FPs were detected. In the case of user-B, 12.6, 21.4, and 38.6 IPs were matched with the daemon access, the LAN+DNS, and the user access whitelists respectively, and no FPs were detected. In the case of user-C, 21.8, 55.4, and 4.8 IPs were matched with the daemon access, the LAN+DNS, and the user access whitelists respectively but 0.4 FPs were detected. Because some web sites set a refresh tag, TCP-HTTP sessions were re-addressed to other web sites after 10 seconds. Consequently, the proposed model can detect the unknown bots with extremely low FPs.

6. Limitations and Considerations

6.1 C&C Servers in Well-known Web Sites

In general, bots communicate with multiple C&C servers. When a bot communicates with only C&C servers located inside well-known web sites listed in the whitelist, FNs of our technique will increase.

6.2 Spam E-mail via Legitimate SMTP Server

Our system cannot detect spam e-mail via a legitimate SMTP server that is listed on the whitelists. Here, 18 bots that are collected in the 44 bots send the spam e-mails. All 18 bots communicate with the C&C server in order to be controlled. Thus, our technique can detect the spam e-mail bots when the outbound packets are sent to the C&C server.

6.3 Suspending P2P Application

There are many P2P applications, e.g., a file exchange, VoIP, and online games, for home users, which communicate with anonymous PCs automatically. When our technique monitors outbound packets, the P2P application must be suspended. Thus, an adaptable area of our technique is for a business domain that restricts P2P applications.

6.4 Maintenance of the Whitelists

Since various kinds of applications are installed in the PCs, it will be complicated to maintenance the whitelists. Effectual maintenance procedures are topics for further study. One possible solution is applying reputation services with whitelists. Where the business environment is concerned, most PCs in a company or a section install the same applications, and thus the daemon access whitelist can be shared with those business PCs.

When the initial PC has been infected or when the bot intrudes in the PC on an initial operation, the whitelits are contaminated with the IP/FQDN of the C&C servers.

6.5 Anti-detection Functions

The bot using a zero-day attack may exhibit a different behavior of network accesses. The bot using a target attack will access new sites one after another to distribute and to download the bot code. Therefore, a blacklist approach will fail to detect those latest bots since it is difficult to keep the blacklist properly. With regard to this issue, no major disruptions are expected in a whitelist approach. So, the infected PC can be detected by the proposed system.

On the other hand, when the bot communicates with the C&C server during the user operation, when the C&C functions are set on legitimate sites, or when the DNS caches are poisoned, the proposed system cannot detect the bot activities. Countermeasures against those anti-detection functions are further study issues.

6.6 Total Cost

The proposed system is installed in each PC without any additional hardware cost, but it requires all PC users to pay the maintenance cost for the whitelist. In contrast, deployment of conventional network-based IDS needs a high performance server, but PC users don't have to maintain any pattern files.

7. Conclusion

We have proposed a bot detection technique that checks outbound packets with destination-based whitelists. The whitelists are described with the IP range and/or superior DN. Because the sizes of the daemon access whitelist that were implemented on three PCs were around 10 lines, it is easy to list up and maintain the legitimate domains. The bot that accesses unknown sites during the non-operating duration can be detected stably. In addition, because our technique can monitor the communication activities of applications and plug-in functions, it is useful for checking on either infection or not-infection.

References

1) Zou, C.C. and Cunningham, R.: Honeypot-Aware Advanced Botnet Construction and Maintenance, *Proc. DSN'06*, pp.199–208 (June 2006).
2) Sudo, T. and Fujiwara, K.: The evaluation of the botnet analysis system based on the virtual Internet environment, *Proc. CSS 2006*, pp.513–158, IPSJ (Oct. 2006).
3) Miwa, S., Miyachi, T., Eto, M., Yoshizumi, M. and Shinoda, Y.: Design Issues of Isolated Sandbox for Analyzing, *Proc. IWSEC 2007*, pp.13–27, IPSJ (Oct. 2007).
4) Kondo, S. and Sato, N.: Botnet Traffic Detection Techniques by C&C Session Classification Using SVM, *Proc. IWSEC 2007*, pp.91–104, IPSJ (Oct. 2007).
5) The Honeynet Project. http://www.honeynet.org/
6) Cyber Clean Center. https://www.ccc.go.jp/en_index.html
7) Goebel, J.: Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation, *Proc. HotBots'07*, USENIX (Apr. 2007).
8) Gu, G., Zhang, J. and Lee, W.: BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic, *Proc. NDSS'08* (Feb. 2008).
9) Gu, G., Porras, P., Yegneswaran, V., Fong, M. and Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation, *Proc. Security'07*, USENIX (Aug. 2007).
10) Cui, W., Katz, R.H. and Tan, W.: Design and Implementation of an Extrusion-based Break-In Detector for Personal Computers, *Proc. 21st ACSAC*, pp.361–370 (June 2005).
11) Liu, L., Chen, S., Yan, G. and Zhang, Z.: Bottracer: Execution-based bot-like malware detection, *Proc. 11th International Conference on Information Security*, pp.97–113, Springer (2008).
12) Nepenthes. http://nepenthes.mwcollect.org/
13) VMware. http://www.vmware.com/
14) FireEye. http://www.fireeye.com/technology/technology_page.php?id=1&keywords=How_the_FireEye_Technology_Works
15) Takemori, K., Isohara, T., Miyake, Y. and Nishigaki, M.: Analysis of Robust Mechanisms into Botnet and Code Set, *Proc. MWS 2008*, Session 2-5, IPSJ (Oct. 2008).
16) MalwareDomainList. http://www.malwaredomainlist.com/hostslist/hosts.txt
17) Akamai. http://www.akamai.com/
18) MacAfee IntruShield. http://www.mcafee.com/us/
19) Proventia. http://www.iss.net/
20) Trendmicro. http://jp.trendmicro.com/jp/home/
21) AVIRA. http://www.free-av.com/

(Received February 1, 2010)
(Accepted January 14, 2011)
(Original version of this article can be found in the Journal of Information Processing Vol.19, pp.169–179.)

Keisuke Takemori received his B.E. and M.E. degrees in Electrical Engineering, and his Ph.D. in Information and Computer Science from Keio University, Japan in 1994, 1996, and 2004. He joined the KDD Corporation in 1996. He is currently a research engineer at the KDDI R&D Laboratories. His current research interests are in network security and smart phone security. He received the Interop Product Award in 2006, the DICOMO Presentation and Paper Awards in 2007, and the MWS Paper Award in 2009.

Takahiro Sakai received his B.I. degree in Informatics from Shizuoka University, Japan in 2009. He is currently working towards the M.I. degree at the Graduate School of Informatics, Shizuoka University, Japan. His research interests are in information security.

Masakatsu Nishigaki received his Ph.D. in Engineering from Shizuoka University, Japan. He served as a Postdoctoral Research Fellow of the Japan Society for the Promotion of Science in 1995. Since 1996 he has been engaged in research at the Faculty of Informatics, Shizuoka University. He is now a Professor at the Graduate School of Science and Technology of Shizuoka University. His research interests are in information security, neural network, circuit simulation, etc.

Yutaka Miyake received his B.E. and M.E. degrees in Electrical Engineering from Keio University, Japan, in 1988 and 1990, respectively. He joined KDD (now KDDI) in 1990, and has been engaged in research on high-speed communication protocol and secure communication systems. He received the Dr. degree in engineering from University of Electro-Communications, Japan, in 2009. He is currently a senior manager of the Information Security Laboratory in KDDI R&D Laboratories Inc. He received the IPSJ Convention Award in 1995 and the Meritorious Award on Radio of ARIB in 2003.