Journal of Computational Science 22 (2017) 301–313

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Analysis of malware download sites by focusing on time series variation of malware

Yasuyuki Tanaka [a,c,*], Mitsuaki Akiyama [d], Atsuhiro Goto [b]

[a] Institute of Information Security (IISEC), 2-14-1 Tsuruyacho, Kanagawa-ku, Yokohama, Kanagawa 211-0835, Japan
[b] Institute of Information Security (IISEC), Japan
[c] NTT Communications Corporation, Japan
[d] NTT Secure Platform Laboratories, Japan

ARTICLE INFO

Article history:
Received 14 September 2016
Received in revised form 24 May 2017
Accepted 28 May 2017
Available online 13 June 2017

Keywords:
Measurement
Analysis
Modeling
Malware
Malware download site

ABSTRACT

As the use of Internet increases, malicious activity has become increasingly problematic. In particular, drive-by download attacks have become a serious problem. As part of an *exploit-as-a-service* ecosystem for drive-by download attacks, malware download sites play a particularly important role. In this study, we analyzed approximately 43,000 malware download URLs to investigate malware distribution and the behavior of malware download sites over an extended period, i.e., over 1.5 years. We found that some sites survive for a very long time and are revived frequently, a finding not revealed in previous research. By focusing on the malware variation, we have identified three categories of malware download sites, i.e., *unchanged*, *every time changed*, *changed occasionally*. We found that 10% of *unchanged* sites survived for more than 500 days, and 10% of *changed occasionally* sites were revived more than 15 times in the entire observation period. We also analyzed sites in terms of IP address changes, anti-virus application results, URL features, and VirusTotal results. We found that each category had different attacker operational and resource characteristics. Finally, based on our findings, we discuss effective countermeasures for each category.

1. Introduction

With the increasing use of Internet, malicious activity has become increasingly problematic, particularly drive-by downloads. Grier et al. described the *exploit-as-a-service* model, a malware distribution ecosystem based on drive-by download attacks [1]. Attackers pay for an exploit kit or service to "do the dirty work" of exploiting a victim's browser. This is easier for attackers than building their own malware distribution network. According to an Internet security threat report from Symantec [2], *exploit-as-a-service* has increased the number of malicious websites, which makes it more difficult for security researchers and engineers to identify and shut down malicious infrastructures.

A typical drive-by download attack involves several malicious websites, i.e., *landing*, *exploit*, and *malware download sites*. A *landing site*, which is often set up on a legitimate site, redirects a victim's web browser to the *exploit site*. Attack code is downloaded from the *exploit site* and executed by a web browser. Then, the web browser forcibly downloads malware from a *malware download site*. *Landing sites* and *exploit sites* are more short-lived than legitimate sites [3,4]. However, based on our experience, *malware download sites* survive longer than *landing* or *exploit sites*. Furthermore, *malware download sites* play an important role in malware infection in the *exploit-as-a-service* ecosystem. Therefore, we observed malware download sites for long periods.

The purpose of this study was to observe malware download sites over a long period and recommend appropriate countermeasures based on an analysis of our observations. Some malware download sites distribute different malware on different days. By observing malware download sites for a long period, we found that such sites can be divided into three categories depending on how frequently the malware changes, i.e., *unchanged* (UNC), *every-time changed* (ETC), *changed occasionally* (COC). For example, for the COC category, in most cases, we obtained malware with the same hash value; however in a very short period, the sites changed the malware and reverted back to the original malware. We found that sometimes the malware was changed to new malware that anti-virus applications cannot detect. In this paper, we first analyze the lifetime and revival peculiarity of malware download sites and

* Corresponding author at: Institute of Information Security (IISEC), 2-14-1 Tsuruyacho, Kanagawa-ku, Yokohama, Kanagawa 211-0835, Japan.
E-mail address: dgs155102@iisec.ac.jp (Y. Tanaka).

http://dx.doi.org/10.1016/j.jocs.2017.05.027
1877-7503/

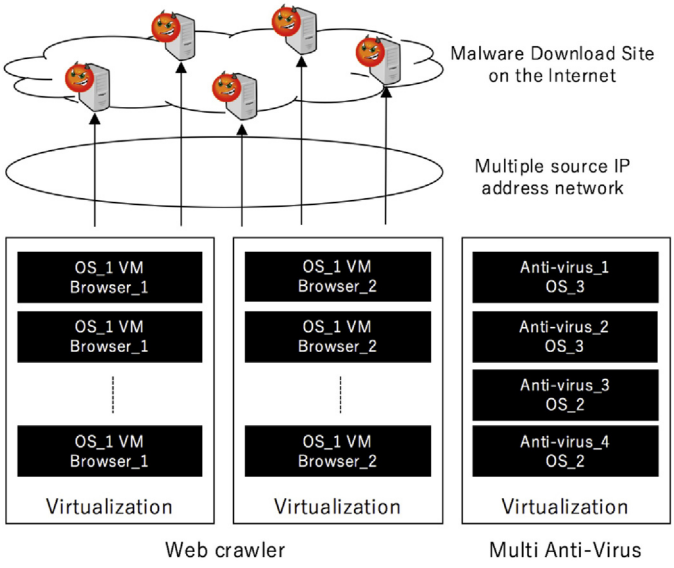

Fig. 1. Monitoring system.

variations in malware or the IP addresses of such sites. We found that some malware download sites live longer and are revived continuously. This reviving behavior differs from that of a benign site. Then, based on our analyzes, we consider the characteristics of the three categories and discuss effective countermeasures for each category. We believe that the analytical results could reveal an attacker behavior and help determine appropriate blacklisting periods for malicious URLs.

We summarize our contributions as follows.

- We examine the lifetime and revival peculiarity of malware download sites over 1.5 years.
- Our analysis shows that, compared to exploit servers, malware download sites have much greater longevity.
- We distinguish three categories by focusing on malware changes and examine the characteristics of each category.
- We consider attacker operational behavior and resources to the three categories. Based on our findings, we discuss effective countermeasures for each category.

2. Environment and dataset

We developed a system to monitor malware download sites. An overview of the system is shown in Fig. 1. In this section, we describe the monitoring system, our observation dataset, and our analytical procedure.

2.1. Checking malware download URLs

Fig. 2 illustrates the procedure used to check malware download URLs. URLs were accessed daily using a high-interaction web crawler that is based on an actual OS and web browser [5], and the activity status of each URL (active or stopped) was recorded. Downloaded files were assessed as malicious or benign using anti-virus applications. Table 1 lists OSs, web browsers, and anti-virus applications that were used. The items in Table 1 correspond to the items labeled in Fig. 1. The four anti-virus applications were selected because they hold top market share in Japan. Some anti-virus applications return false negatives; however, top market share anti-virus applications return fewer false positives than less popular applications. We also discuss our four anti-virus application choice in Section 4.2. Note that the signatures of anti-virus applications were always updated to the latest version.

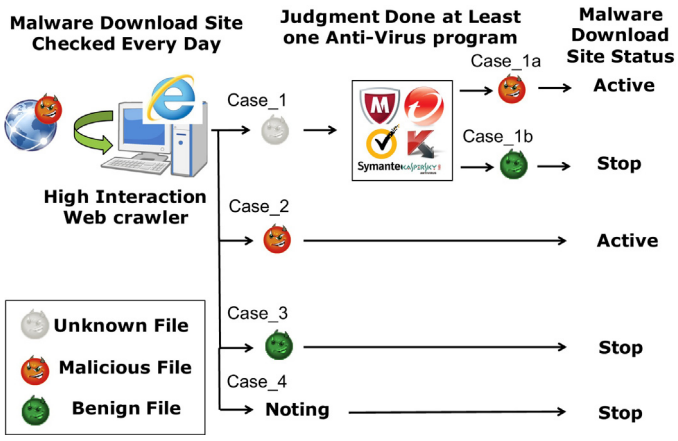

Fig. 2. Checking for malware download URL.

Table 1
Experimental settings.

Item	Product and version
Virtualization	VMware ESXi 5.5
OS_1 (Web crawler)	Windows XP SP2
OS_2 (Anti-virus)	Windows 7 Professional SP1
OS_3 (Anti-virus)	CentOS 6.3
Browser_1 (Web crawler)	Internet Explorer 6
Browser_2 (Web crawler)	Internet Explorer 8
Anti-virus_1	Trend Micro ServerProtect for Linux [6]
Anti-virus_2	Symantec Endpoint Protection [7]
Anti-virus_3	McAfee Endpoint Protection Suite [8]
Anti-virus_4	Kaspersky Security [9]

Table 2
Status of four representative URLs.

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	√	√	√	×	√	√	×	√	×	√	√	√
URL_2	×	√	×	×	×	√	×	×	×	×	×	×
URL_3	√	√	√	×	√	√	×	×	√	√	√	×
URL_4	×	×	√	√	√	×	√	√	√	√	×	×

Our web crawler accesses a URL and downloads a file. If the file was downloaded for the first time, it was called an unknown file (Fig. 2, Case_1). The unknown file was scanned by the four anti-virus applications. If at least one anti-virus application identifies the file as malicious, the source URL's status was set to active (Fig. 2, Case_1a), otherwise, it was set to stop (Fig. 2, Case_1b). This heuristic threshold is based on the four selected anti-virus applications, which we discuss in Section 4.2. If the web crawler accessed the same URL, it sometimes retrieved a known hash file. If we received a known malicious or benign file, the URL status was set to active or stop (Fig. 2, Case_2 and Case_3, respectively). If an error, such as HTTP or DNS errors, occurred or no file was retrieved, the URL status was set to stop (Fig. 2, Case_4). We used multiple IP addresses for the web crawler to prevent attackers from detecting our observations.

We observed and recorded the status of these URLs every day for 1.5 years. Table 2 shows the status of four representative URLs over 12 days. The status of URL_1 was stop on days 4, 7, and 9, and URL_2 was active on days 2 and 6. The malware hash values are shown in Table 3. These hash values were generated by the linux OS `sha1sum` command. In this example, URL_1 and URL_2 always yielded the same malware; however, different malware was obtained from URL_3. The malware from URL_4 also changed, but in an irregular manner. The IP addresses of the URLs are listed in

Table 3
Malware hashes of four representative URLs.

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	A	A	A		A	A		A		A	A	A
URL_2		B				B						
URL_3	C	D	F		G	H			I	J	K	
URL_4			O	O	O		P	O	O	O		

Table 4
IP addresses of four representative URLs

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	a	b	a		a	b		b		a	b	a
URL_2		a				a						
URL_3	c	d	e		f	g			h	i	j	
URL_4			c	c	c		b	c	c	c		

Table 5
Top 10 NetName details.

DAY	1	2	3	4	5	6	7	8	9	10	11	12
URL_1	a	b	a		a	b		b		a	b	a
URL_2		a				a						
URL_3	c	d	e		f	g			h	i	j	
URL_4			c	c	c		b	c	c	c		

Table 4. The IP address was consistent for URL_2; however, the IP address for URL_3 varied frequently.

In this study, we focused on malware download sites; therefore, we only analyzed *malware download site* URLs. Using the common vulnerabilities and exposures (CVE) number detection function of the high-interaction web crawler [5], we excluded *exploit sites*.

2.2. Data characterization

Our observation period covered 607 days from February 1, 2014 to September 30, 2015 (Fig. 3). In addition to continuous observation of known malicious URLs, we also carried out web crawling to discover unknown malicious URLs. The reason for crawling unknown malicious URLs is that we were not able to obtain valuable observation data since some malicious URLs are short-lived and disappeared quickly; therefore, we had to add new malicious candidate URLs to our experiment. Fig. 3 shows that the cumulative number of new malicious URLs increased during our observation period. The number of URLs reached 43,034 on March 20, 2015 and we stopped crawling to discover new malicious URLs at that time (Fig. 3, red line). In other words, we observed all URLs for at least 195 days (approximately half a year), from March 20, 2015 to September 30, 2015. Half of these URLs (21,517 URLs) were observed for 505 days, from May 14, 2014 to September 30, 2015 (Fig. 3, dashed green line). These malicious candidate URLs were obtained from open-feeds sources. Representative open-feeds include the Malware Domain List [10], urlQuery [11], and CLEAN MX [12]. These candidate URLs were checked using the procedure outlined in Fig. 2. Fig. 3 also shows that the cumulative number of disappeared malicious URLs during our observation period (dashed gray line). Note that, here, *disappeared* indicates a URL that did not become active again until the end of the observation period, September 30, 2015.

Table 5 shows the top 10 autonomous systems (ASs) relative to the 43,034 collected URLs. Web searches revealed that most ASs listed in this table provide cloud-hosting services. A similar finding was reported in a previous study [4]. We also show each AS's ranking with the Center for Applied Internet Data Analysis (CAIDA) ranking [13]. Previous research [4] showed that the number of malicious sites is not proportional to the scale of the AS but tends to converge to a specific AS. Similar tendencies were observed in our dataset. It also indicates that our dataset is not unreliable. Note that some of the NetNames have multiple autonomous system numbers (ASNs), and some ASNs have multiple country codes (CCs) referring to multiple country prefixes.

Table 6
Breakdown of three categories.

Category	Count
UNC	11,155
ETC	15,714
COC	16,165
Total	43,034

2.3. Analytical procedure

The same malware download site can sometimes distribute different malware on different days. By observing malware download sites for a long period, we found that such sites can be divided into three categories depending on the change of malware. We now examine the relationship of these three categories and attacker resources. We call the first category *unchanged* (UNC). Whenever we access this type of URL, we always obtain the same hash malware. We call the second category *every-time changed* (ETC). Whenever we access this type of URL, we always obtain different malware. We call the third category *changed occasionally* (COC). We sometimes obtain different hash malware from this type of URL. Figs. 4–6 show the time series variation of malware for UNC, ETC, and COC URLs, respectively. The X-axis shows time, and a single bar represents seven days. The Y-axis shows the number of detected anti-virus application results. The height of the Y-axis depends on the status of the malware download site (Section 2.1). For COC, in most cases, we obtained malware with the same hash value; however, in a very short period, the website changed the malware and reverted to the original malware (Fig. 6, light blue) immediately. Since, in our experience, these different malware from COC are often uncommon in anti-virus programs and have not been submitted to VirusTotal [14], we investigated this point.

First, we divided the 43,034 URLs into the three categories. Table 6 shows the classification results of our dataset. Next, we selected features for each URL to estimate the lifetimes and resources of URLs, such as IP address variation. We compared the three categories relative to these features. The definitions of the features used for lifetime analysis and the results are described in Section 3. The definitions of the features used for behavior analysis and the results are described in Section 4. Finally, Section 6 discusses the characteristics of the three categories based on these analyzes.

3. URL Lifetime Analysis

In this section, we examine the lifetime and revival peculiarity of *malware download sites* relative to the three categories described in Section 2.3. These analytical results could help determine appropriate blacklisting periods for malicious URLs. We explained how victims are infected with malware in Section 1. As mentioned previously, *malware download sites* play a more important role in malware infection in the *exploit-as-a-service* ecosystem compared to *landing* or *exploit sites*. For example, a single *malware download sites* can be linked and used by many *landing* or *exploit sites*. Therefore, *malware download sites* must stably provide malware for *landing sites* and/or *exploit sites* over a long period. Our observation focused on the behavior of malware download sites over long periods to reveal how malware is distributed over a long period. We found through that some malware download sites live longer and

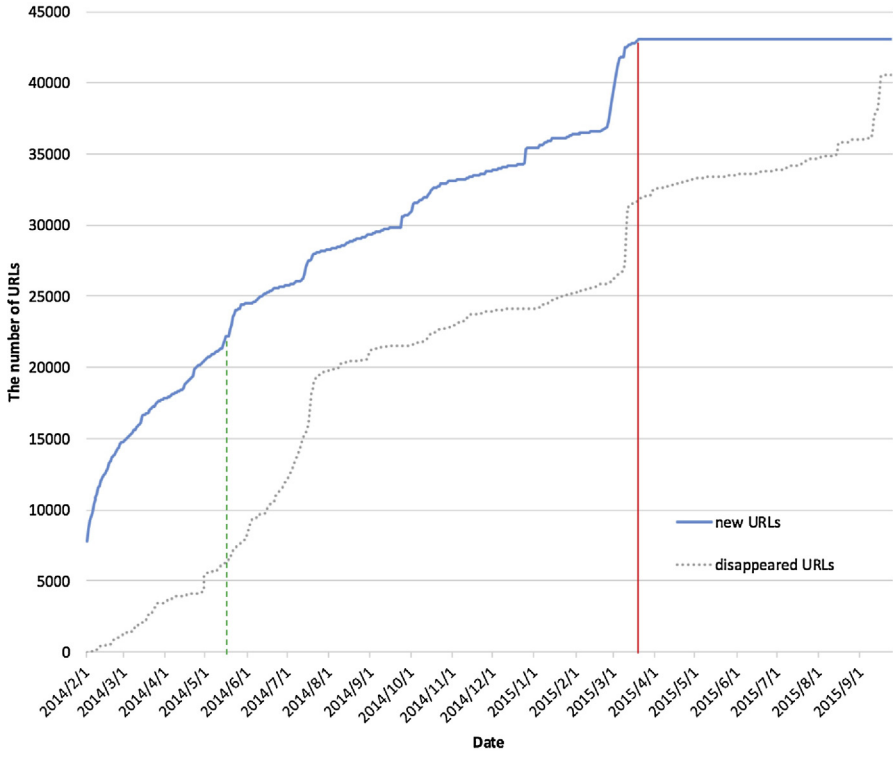

Fig. 3. Cumulative number of increased new and disappeared malicious URLs.

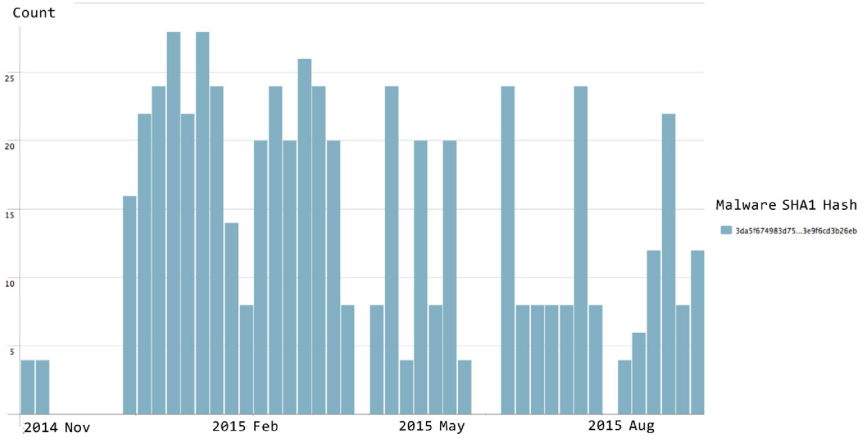

Fig. 4. UNC URL.

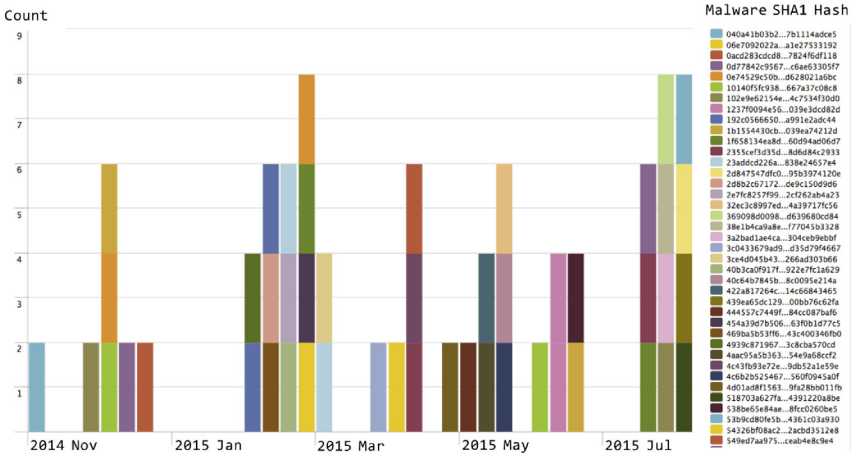

Fig. 5. ETC URL.

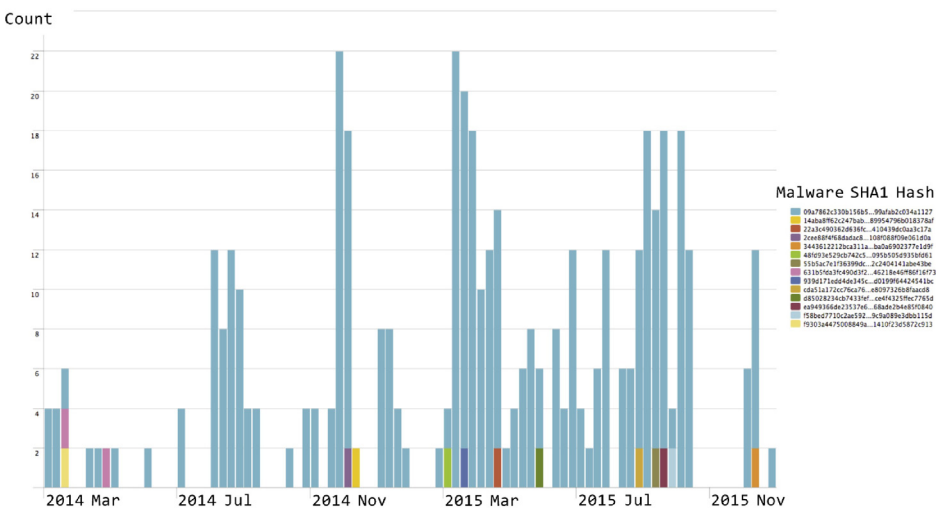

Fig. 6. COC URL.

are revived continuously, and this reviving behavior differs from that of benign sites.

3.1. Feature definitions

Lifetime. We define *Lifetime* as the period between the first and last observation days for each URL. For example, as shown in Table 2, the *Lifetime* of URL_2 and URL_4 is five and eight days, respectively. We do not include the in-between status in this definition. Note that, in Table 2, the example observation period was only 12 days, whereas our actual observation period was approximately 1.5 years.

Active days. We define *Active days* as the total number of *Active days* for each URL. For example, the number of *Active days* for URL_2 and URL_4 is two and seven (Table 2), respectively. In contrast to *Lifetime*, we do include the in-between status in this definition.

IntervalMax and IntervalMin. First, we define Interval as the number of days with continuous stop status. *IntervalMax* is the maximum Interval value. In Table 2, the *IntervalMax* of URL_2 and URL_3 is three days and two days, respectively. If we cannot observe active status over observation period, we do not regard it as continuous stop status. The *IntervalMax* of URL_2 is not six days, from day 7 to day 12. The *IntervalMax* of URL_2 is three days, from day 3 to day 5, in this definition. We also define *IntervalMax2* and *IntervalMax3* as the second and third maximum Interval values, respectively, and *IntervalMin* is the minimum Interval value.

IntervalAve and IntervalStd. IntervalAve is the mean value of each Interval, and *IntervalStd* is the variance of each Interval value.

Stop5 and Stop10. We define *Stop5* and *Stop10* to analyze each URL's revival behavior. *Stop5* is the number of Intervals whose value is greater than five days and less than ten days, and *Stop10* is the number of Intervals whose value is greater than ten days. For example as seen in Table 2, *Stop5* is zero and *Stop10* is zero for any of the URLs.

3.2. Analysis results

Lifetime. Fig. 7 shows the cumulative distribution function (CDF) for *Lifetime* and *Active days*. The light green, light red, and light blue lines show the *Lifetime* for UNC, ETC, and COC, respectively. As can be seen, the mean values are 203.3, 68.3, and 195.8 days for UNC, ETC, and COC, respectively. Obviously, UNC and COC were longer-lived than ETC. In particular, 10% of UNC URLs (light green) survived for more than 500 days. A previous study [4] showed that 10% of exploit servers survived for more than one week, 5% of exploit servers survived more than two weeks, and some servers survived for up to 2.5 months. Compared to this conventional result, we revealed that malware download sites have much greater longevity. UNC and COC have similar graphs; however, the trend changes at approximately 130 days. At less than 130 days, COC URLs was longer-lived than UNC, but after 130 days, UNC became longer-lived than COC.

Table 7
Average *IntervalMax* and *IntervalMin*.

Category	*IntervalMax*	*IntervalMax2*	*IntervalMax3*	*IntervalMin*
UNC	16.42	7.699	5.706	1.549
ETC	15.67	6.303	3.864	2.054
COC	20.25	9.810	7.034	0.9053

Table 8
Average *IntervalAve* and *IntervalStd*.

Category	*IntervalAve*	*IntervalStd*
UNC	2.571	2.343
ETC	3.460	2.979
COC	2.163	3.103

Active days. The dark green, dark red, and dark blue lines in Fig. 7 show the *Active days* for UNC, ETC, and COC, respectively. The mean values are 120.9, 26.55, and 105.1 days for UNC, ETC, and COC, respectively. As can be seen, UNC and COC were longer-lived than ETC. In particular, 10% of UNC (dark green) was active for more than 290 days. Overall, *Active days* (Fig. 7, dark line) shows the same tendency as *Lifetime* (Fig. 7, light line), and *Active days* is present to the left of *Lifetime*.

IntervalMax and IntervalMin. Table 7 shows the mean values of *IntervalMax*, *IntervalMax2*, *IntervalMax3*, and *IntervalMin* for each category. COC show higher *IntervalMax*, *IntervalMax2*, and *IntervalMax3* values and a lower *IntervalMin* value than the other two categories. The idle period for COC was long (i.e., *IntervalMax* was high); however, once started, it worked continuously without rest (i.e., *IntervalMin* was low). The *IntervalMin* value of COC is less than one day because our URL check interval was sometimes slightly less than a strict 24 hour period.

IntervalAve and IntervalStd. Table 8 shows the mean values of *IntervalAve* and *IntervalStd*. For IntervalAve, the COC values are mostly small, i.e., 0.63 times those of ETC. This suggests that COC was used intensively at one time, as mentioned in Section 3.2. In contrast, the activity of ETC was sparse.

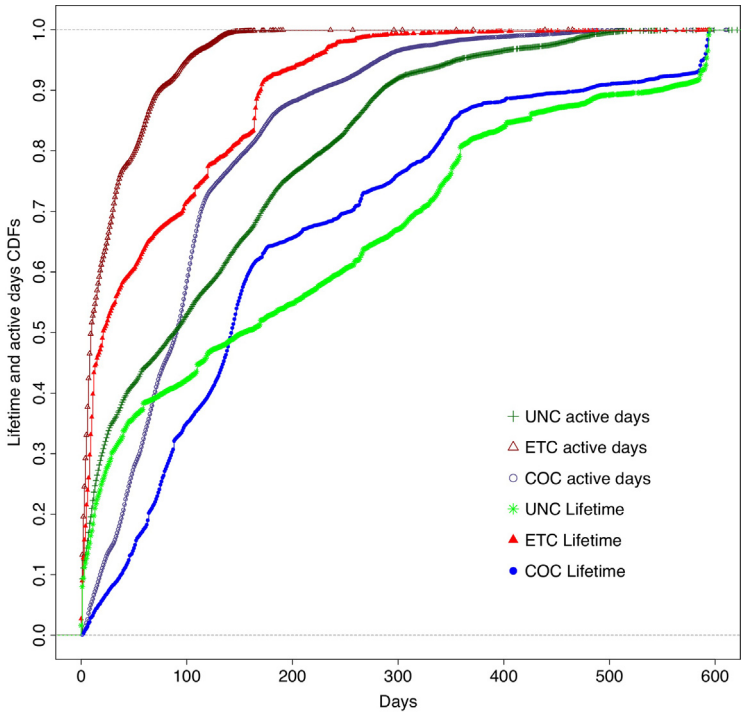

Fig. 7. *Lifetime* (light lines) and *Active days* (dark lines).

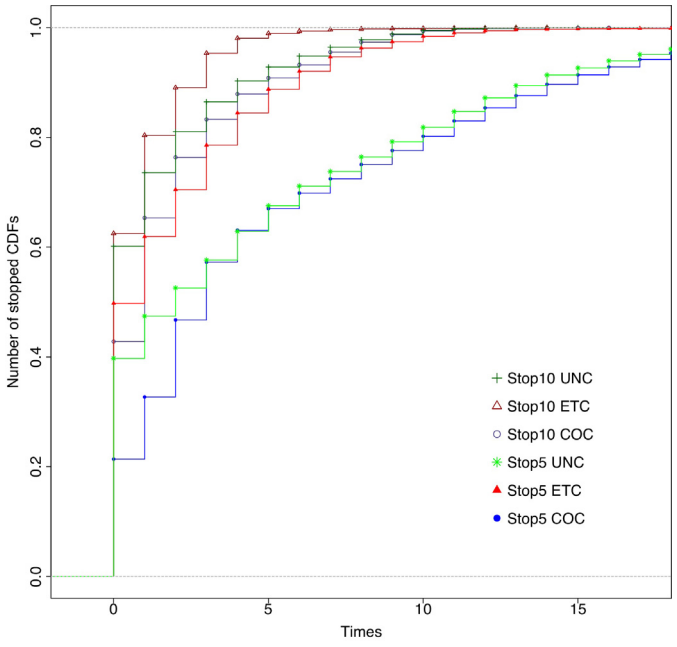

Fig. 8. *Stop5* (dark lines) and *Stop10* (light lines).

Stop5 and Stop10. Fig. 8 shows CDF for *Stop5* and *Stop10*. The light line is *Stop5* result. The mean values of *Stop5* are 4.68, 1.90, and 5.29 for UNC, ETC, and COC, respectively. Overall, UNC and COC show similar curves, and these curves are larger than the ETC curves. In particular, 10% of COC (light blue) were revived more than 15 times. The maximum number for the COC URLs was 32 times. The dark line is the *Stop10* result. The mean values of *Stop10* were 1.29, 0.77, and 1.70 for UNC, ETC, and COC, respectively. Overall, *Stop10* was less than *Stop5*. In particular, 5% of COC (dark blue) were revived more than eight times, and half of them more than three times.

4. URL behavior analysis

In this section, we examine variations in the downloaded malware or IP addresses of malware download sites. These analytical results reveal an attacker's behavior and contribute to using malicious URL blacklisting more effectively. For example, URL blacklisting is particularly effective for the COC (*changed occasionally*) category because, in most cases for this category, we obtained malware with the same hash value. However such sites immediately changed malware and reverted to the original malware. In our analysis, this changed malware sometimes represented new malware that anti-virus applications cannot detect. In this case, we consider that URL blacklisting priority should be high.

4.1. Feature definitions

UniqIP. We define *UniqIP* as the number of unique IP addresses of each URL. For example, as can be seen in Table 4, the *UniqIP* values of `URL_1` and `URL_3` are two and eight, respectively.

UniqIPMax and UniqIPMin. We define *UniqIPMax* as the maximum number of unique IP addresses of each URL, and *UniqIPMin* is the minimum number of unique IP addresses. For example, as shown in Table 4, the *UniqIPMax* values of `URL_1` and `URL_3` are five and one, respectively.

UniqIPAve and UniqIPStd. We define *UniqIPAve* as the mean value of the number of individual unique IP addresses and *UniqIPStd* as the variance in the number of individual unique IP addresses.

IPEntropy. We define *IPEntropy* in terms of the Shannon entropy of an IP address as follows:

$$IPEntropy = -\sum_{i \in IPaddress} P(i) log P(i)$$

AVRate. Note that *AVRate* is not a feature of IP addresses. We define *AVRate* as the proportion of the number of detected anti-virus signatures. A high *AVRate* indicates that many anti-virus applications have many signatures; thus, such URLs possibly

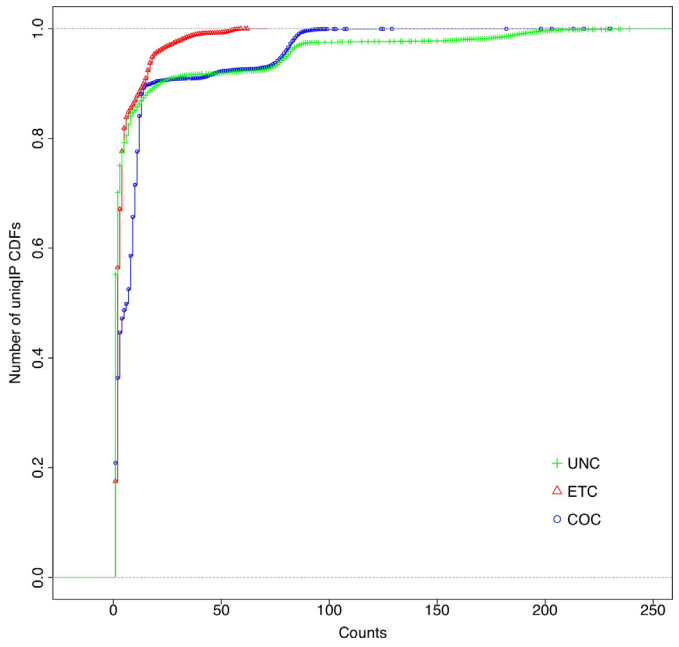

Fig. 9. UniqIP.

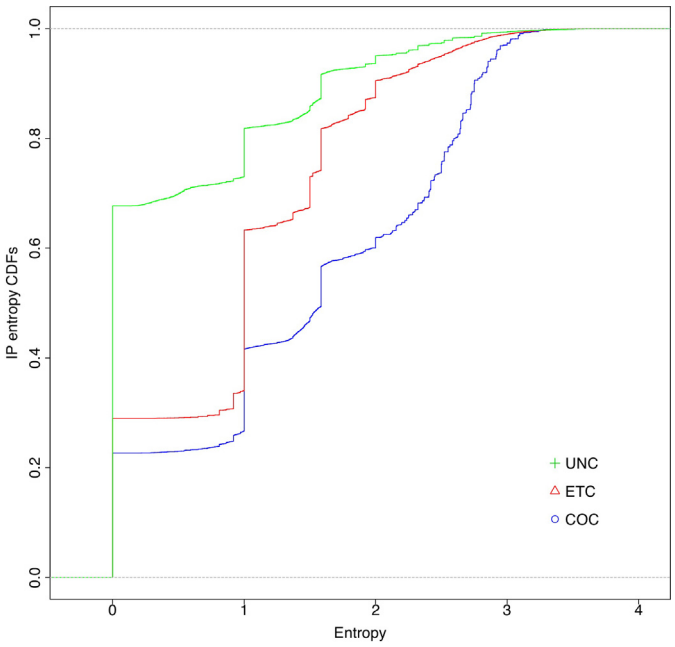

Fig. 10. IPEntropy.

Table 9
Average *UniqIPMax*, *UniqIPMin*, *UniqIPAve*, and *UniqIPStd*.

Category	*UniqIPMax*	*UniqIPMin*	*UniqIPAve*	*UniqIPStd*
UNC	75.63	61.82	67.56	5.876
ETC	12.03	6.527	8.801	2.264
COC	52.88	29.99	37.94	8.655

provide known malware. A low *AVRate* indicates that anti-virus applications have few signatures; thus,such URLs possibly provide unknown malware.

Netloc, Path, and Query. Note that *Netloc*, *Path*, and *Query* are not features of a time series. We define these features to investigate URL features. *Netloc* is the length of the host part of each URL, and *Path* and *Query* are the lengths of each URL, respectively.

VTsubmit and VTscore. Note that these are also not features of a time series. In malware analysis, VirusTotal [14] is commonly used as a maliciousness indicator. VirusTotal gives scan results for over 50 different anti-virus applications. As VirusTotal acts as a huge malware intelligence database, the information whether VirusTotal know is valuable. We can use such information as an indicator. In addition to our *AVRate* feature, by adding analyzes using VirusTotal metadata, we consider that higher quality analysis results can be obtained. We obtained two results by searching each malware hash value. We define *VTsubmit* as whether the malware has been submitted to VirusTotal. If the malware has been submitted, the *VTsubmit* value is one; otherwise, the *VTsubmit* value is zero. The other result is the number of anti-virus applications detected as malware, which we define as *VTscore*.

4.2. Analysis results

UniqIP. Fig. 9 shows CDF for *UniqIP*. The mean values are 11.8, 5.10, and 12.2 for UNC, ETC, and COC, respectively. This shows that the number of unique IP addresses used in ETC was less than those of UNC and COC. In particular, 2% of UNC used more than 180 IP addresses during our 1.5-year observation period.

UniqIPMax and UniqIPMin. Table 9 shows *UniqIPMax* and *UniqIPMin* results. In addition to the discussion in Section 4.2, this also shows that the number of unique IP addresses used for ETC was less than of UNC and COC. However, we could not obtain significant consideration relative to *UniqIPMax* or *UniqIPMin*.

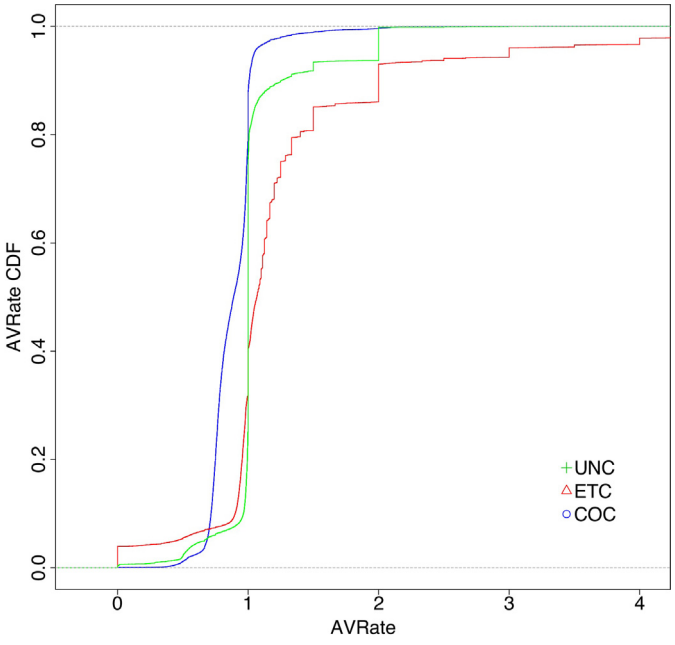

Fig. 11. AVRate.

UniqIPAve and UniqIPStd. As can be seen in Table 9, *UniqIPMax*, *UniqIPMin*, and *UniqIPAve* are greater for UNC compared to COC. However, *UniqIPStd*, for UNC is less than that of COC. This indicates that COC's IP addresses were frequently variable despite not having large IP address resources compared to UNC.

IPEntropy. Fig. 10 shows CDF for *IPEntropy*. The mean values are 0.456, 1.02, and 1.50 for UNC, ETC, and COC, respectively. Overall, COC shows more entropy than the other two categories. This indicates that its variation was high, as discussed as in Section 4.2.

AVRate. Fig. 11 shows the CDF for *AVRate*. The mean values are 1.05, 1.42, and 0.885 for UNC, ETC, and COC, respectively. Overall, COC shows smaller *AVRate* than the other two categories. According

Table 10
Average *Netloc*, *Path*, and *Query*.

Category	*Netloc*	*Path*	*Query*
UNC	19.77	38.05	13.97
ETC	25.64	30.99	247.2
COC	20.36	27.71	31.98

Table 11
Average *VTsubmit* and *VTscore*.

Category	*VTsubmit*	*VTscore*
UNC	0.960	30.5
ETC	0.00265	33.0
COC	0.142	27.3

Table 12
Unsubmitted rate of each observation time for COC.

Observation times	Unsubmitted rate
1	94.0%
2–10	29.3%
Over 10	2.43%

to the definition of *AVRate*, it is highly possible that COC provides more unknown malware than the other two categories.

Netloc, Path, and Query. First, we investigate the association using the exploit kit from the URL features. For example, BlackHole is known to use a URL with specific parameters. Grier et al. showed this as `w.php(.*?)&e=(.*?)` in their work [1]. In addition, according to [15], we also found `main.php&main=`. We could obtain URL features relative to current popular exploit kits, such as Angler, RIG, NUCLEAR, and FIESTA [16]. We attempted to identify exploit kits with these features. In summary, part of our dataset was created using BlackHole; however, we could not assess the categories of other popular exploit kits. Thus, we conclude that it is difficult to identify the type of exploit kit using only URL features. Note that Grier et al. attempted to identify exploit kits using URL features; however, they could not identify some categories. Table 10 shows the mean values for *Netloc*, *Path*, and *Query*. Little difference is observed relative to *Netloc* and *Path*; however, we observe significant difference relative to *Query*. Note that the ETC values are approximately ten times greater than those of the other two categories.

VTsubmit and VTscore. Since VirusTotal has a request limit of at most four requests/minute, we randomly selected 50 URLs from each category. We obtained VirusTotal results by searching each malware hash value. The numbers of unique malware were 50, 755, and 1253 for UNC, ETC, and COC, respectively. Table 11 shows the average *VTsubmit* and *VTscore* values for each category. The *VTsubmit* value of UNC was the greatest. In particular, 96.0% of malware distributed from UNC was submitted to VirusTotal in the past; thus, VirusTotal was already aware of such malware. The *VTsubmit* value for ETC was the lowest. We obtained different malware each time; thus, as a result, there was less probability that these malware had been submitted to VirusTotal. In addition, the *VTscore* for ETC was the greatest. This indicates that, if malware from ETC was submitted, they were known to anti-virus applications. The high *AVRate* value of ETC also shows that such malware were familiar.

Next, we investigated malware from COC. For the time series from COC, in most cases, we obtained malware with the same hash value; however, we obtained changed malware within a very short time, as described in Section 2.3. Such different malware from COC are often unommon in anti-virus programs and have not been submitted to VirusTotal; thus, we investigated this point. Table 12 shows the unsubmitted rate for each observation days. When the number of observation days was only one, the unsubmitted rate

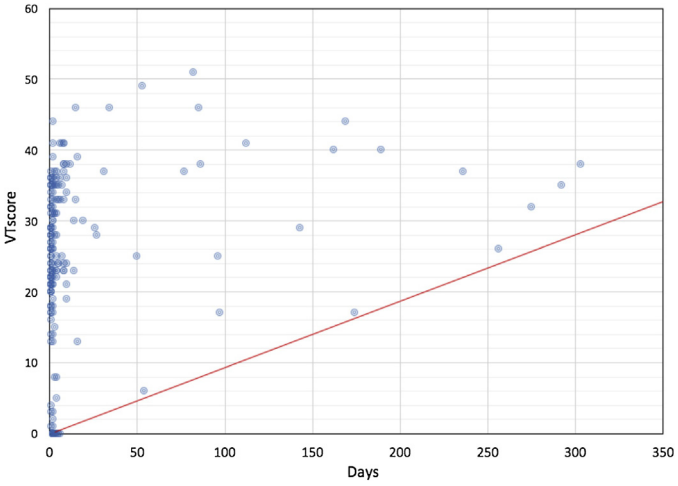

Fig. 12. VTScore.

was 94.0%. This shows that such changed malware tended not to be submitted to VirusTotal (Table 12). As the number of observation days increased, the unsubmitted rate decreased. This indicates that these changed malware were unfamiliar; thus, we assume that attackers may use them for special purposes, such as targeted attacks. Fig. 12 shows the associations between the number of observation days and *VTscore*. As the number of observation days increased, *VTscore* tended to increase (Fig. 12, red line). Less than 10 days accounted for 80% of the total, and 30% of those had a *VTscore* of less than 20. Thus, it is necessary to pay attention to such discreet malware.

Furthermore, we analyzed metadata from VirusTotal. Fig. 13 shows each anti-virus application result for COC malware. Here, "Positive" indicates that the anti-virus software judged malicious files, and "Negative" indicates that the anti-virus software judged benign files. This information is shown in descending order of the number of positives. As the anti-virus applications we used (Fig. 1 and Table 1) are in the upper ranks (No.1 and No.4) and middle ranks (No.26 and No.38), we found that the anti-virus product choice of our monitoring system and our threshold (Section 2.1) were appropriate. On the other hand, Fig. 13 shows that there is a product group with an exceedingly high ratio of negatives than positives. Whether a positive result is a true positive or a false positive is unclear, and whether a negative is a true negative or a false negative is also unclear. Nevertheless, these results suggest that there are two product groups, i.e., exceedingly high positive, and exceedingly high negative. However in the *VTscore* definition we only use positive values; thus, it may be possible to use negative values as an indicator in future. Finally, Fig. 13 also shows that groups wherein both positive and negative results are exceedingly low work poorly as anti-virus application.

5. Consideration of categorization automation

In this section, we consider automation of categorization. In this study, we analyzed our three category definition (Section 2.3). We performed the following experiments to verify whether automatic classification finds other significant classifications. We attempted to categorize our dataset (43,034 URLs; 1.5-year observation period. (Section 2.2)) using clustering method with the features defined in Section 3.1 and 4.1. We could make some groups using a clustering method; however, we could not find any significant characteristics of malware download sites from these each groups. This indicates that it is difficult to find automatically these heuristics such as our three category which we found.

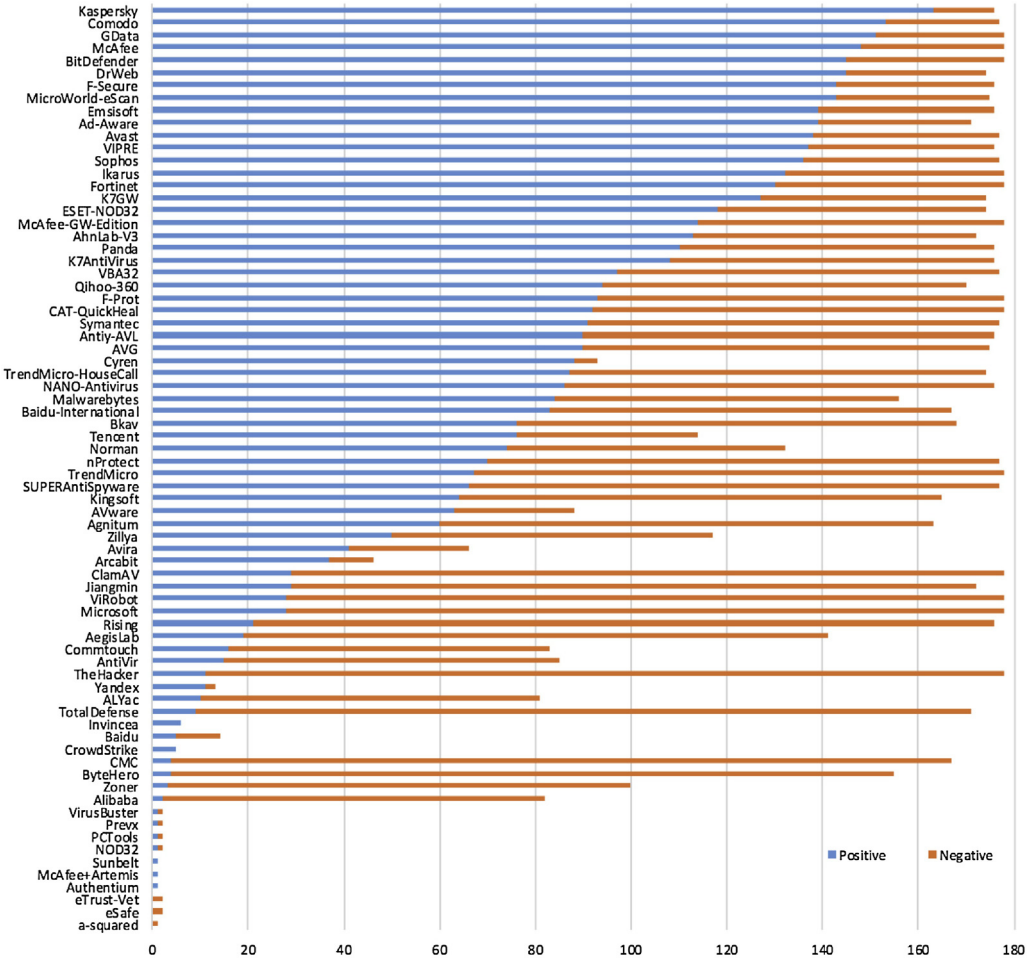

Fig. 13. Anti-virus application results.

Table 13
Feature importance.

Feature	Value	Feature	Value
Lifetime	464.8	UniqIP	331.3
Active days	339.6	UniqIPMax	364.8
IntervalMax	135.9	UniqIPMin	179.3
IntervalMax2	138.0	UniqIPAve	459.9
IntervalMax3	134.7	UniqIPStd	410.2
IntervalMin	1.855	IPEntropy	321.4
IntervalAve	288.3	AVRate	1882
IntervalStd	255.7	Netloc	415.9
Stop5	97.29	Path	668.5
Stop10	56.56	Query	308.3

Accordingly, to further investigate automation of categorization, we attempted to automatically identify our three categories using machine learning. If we can predict the three categories, we can define and perform countermeasures, as discussed in Section 7.1 and 7.2. Here, we also use the features defined in Section 3.1 and 4.1. We selected a random forest algorithm for machine learning because we can obtain each feature importance easily with this method. First, we used all data from our observation period (607 days from February 1, 2014 to September 30, 2015). The identification error rate obtained with the random forest algorithm was 11.16%. Table 13 shows the importance of each feature. As can be seen, the AVRate has the greater value, followed by Path, Lifetime, and UniqIPAve. The features of these high value contribute to identification. Note that we did not use *VTsubmit* and *VTscore* features because we cloud not obtain data for our all URLs (Section 4.2). Next, to consider the relationship between the observation period and identification accuracy, we then recalculated identification error rate by changing the observation period to 14, 28, 59, 89, 120, ⋯, and 607 days. Fig. 14 shows the identification error rate and observation periods. As the number of observation days increased, the error rate decreased proportionally. As a benchmark for an appropriate observation period, approximately three years would be required to achieve a 5% error rate.

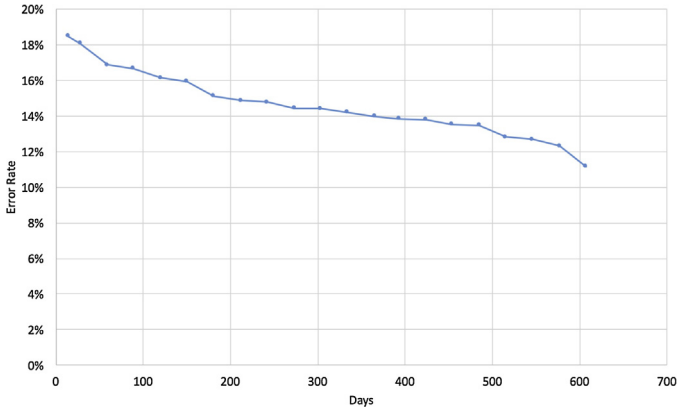

Fig. 14. Identification error rate and observation periods.

6. Characteristics of each category

In Section 3, URL lifetime analysis, we revealed the lifetime, revival peculiarity, activity interval based on the ten defined features. In Section 4, URL behavior analysis, we revealed the characteristics of IP address changes and the characteristics of distributed malware based on the twelve defined features. Here, we consider the characteristics of the three categories, i.e., UNC, ETC, and COC, based on ground truth, the URL lifetime analysis mentioned in Section 3 and URL behavior analysis mentioned in Section 4.

6.1. UNC characteristics

Longevity. UNC had the highest *Lifetime* and *Active days* values, as shown in Section 3.2. In particular, 10% survived more than 500 days.

Substantial IP resources but little change. As shown in Section 4.2, UNC demonstrated substantial IP address resources. In particular, 2% used more than 180 IP addresses during our approximately 1.5-year observation period. However, their variation was not high because their *IPEntropy* was low, as shown in Section 4.2.

6.2. ETC characteristics

Short-lived. The *Lifetime* and *Active days* values of ETC were lower than those of the URLs in the other two categories, as shown in Section 3.2.

Sparse activity. Although ETC was short-lived,their activity was sparse in their active period, in other words, compared with those in the other two categories, they stopped and started continually because their *IntervalAve* and *IntervalMin* were high and their *Stop5* and *Stop10* were low, as discussed in Section 4.2.

Fewer IP resources but more changes. As shown in Section 4.2, ETC had fewer IP address resources than those in the other two categories, but their variation was high because their *IPEntropy* was higher than those of UNC.

Distribute common malware. The *AVRate* value of ETC was higher than those of the URLs in the other two categories, as shown in Section 4.2. Malware for which many anti-virus vendors have signatures is popular and known; thus, we conclude that ETC URLs provide known, and in fact, obviously common malware.

Long Query part. The *Query* value of ETC was 10 times larger than those of the URLs in the other two categories. In Section 4.2, we stated that we could not identify a specific exploit kit's name, but we believe that a long URL query path indicates that there is a high possibility of some malware distribution platforms. We discussed this in Section 7.1.

6.3. COC characteristics

Longevity. COC had high *Lifetime* and *Active days* values, as shown in Section 3.2. In particular, 10% survived more than 460 days. However, the percentage of URLs that survived longer than 130 days was less than that of UNC.

Intensive activity. COC was used intensively at one time, as shown in Section 3.2. Once they entered the active state, they continued to work for a certain period. This trend was opposite the sparse activity of ETC.

Revived many times. As shown in Section 3.2, COC revived more often than those in the other categories. In particular 10% revived more than 15 times. The maximum number of revivals was 32 times during observation.

Substantial IP resources and more changes. As shown in Section 4.2, COC had substantial IP address resources, similarly to UNC, but their variation was high because their *IPEntropy* value was much higher than those of UNC.

Distributes uncommon malware. The *AVRate* value of COC was lower than those of the other two categories, as shown in Section 4.2. Malware for which many anti-virus vendors have only a few signatures is unpopular and unknown; thus, we conclude that COC URLs provide unknown, i.e., uncommon malware. The low *VTscore* of COC also shows that they distributes more uncommon malware than UNC or ETC.

Changes to unsubmitted malware and return. As shown in Section 4.2, COC changed unsubmitted malware in a very short period and returned to the originally submitted malware.

7. Discussion

7.1. Considerations for mitigation

In this section, we first infer the the operations and resources of attackers for each category based on ground truth described in Section 6. Then, we discuss effective countermeasures for each category.

UNC. We considered these to be sites affected by malware as a result of the vulnerabilities of regular sites or general user blogs. *Longevity* and *substantial IP resources but little change* characteristics are considered to have taken over the properties of a parasitized site. Site administrators are commonly unaware of the presence of malware. The attacker also does not have any interest in the malware operation of the site. Although this is the case in instances we observed, a web site sometimes moves one AS to another AS as business demands. Therefore, the IP address range of the fully qualified domain name (FQDN) changes without any intention of the attacker.

A fundamental countermeasure is to take down a malware download site; however, the operational cost of this is high. Using the longevity feature, we think that an effective countermeasure for the UNC category would be URL blacklisting. This would be effective because the URL is not changed even if the AS or IP address is changed. URL blacklisting for UNC is also effective in terms of operating cost and over-blocking. We discuss this in Section 7.2.

ETC. We infer that there are probably exists some malware distribution platform that changes malware every time. We believe this is slightly different from a typical exploit kit for two reasons. First, generally, the primary intention of an exploit kit is to execute malicious code using various vulnerabilities and downloaded malware [2,15,16]. Note that, for a typical exploit kit, it does not matter how malware is distributed or how often malware is changed. Second, as mentioned in Section 2.1, using a high-interaction web crawler [5], we excluded exploit sites URLs from our dataset. This means that if people access ETC URLs using a vulnerable browser, no exploits occur, i.e., only malware downloads occur.

We infer that the characteristics of *Sparse activity* and *Less IP resources but more changes* depend on such platforms. For example, a malware distributor uses IP resources as much as possible to avoid being blacklisted, and once a URL is blacklisted, it drops that specific URL. Thus, it may become *Short-lived*. In addition, such platforms may provide common rather than uncommon malware.

To mitigate this, we must conduct more research into malware distribution platforms and improve identification technology. These platforms may be a subspecies of exploit kits or a new type that only distributes different malware without using vulnerabilities used by typical exploit kits. To the best of our knowledge, no relevant studies have been conducted to evaluate this idea.

COC. In terms of malware changes, COC may also involve malware distribution platforms observed in connection with ETC. According to our data, COC has common characteristics with UNC,

i.e., *Longevity* and *Substantial IP resources*. Thus, we think that the COC infrastructure is similar to that of UNC. However, its operation differs, as demonstrated by the *Intensive activity*, *Substantial IP changes*, and *Revived many times* characteristics. We hypothesize that the operation of COC is sufficiently advanced to distribute uncommon and unsubmitted-to-VirusTotal malware to targets accurately with specific timing. This may be used in targeted attacks.

Using the longevity and revival many times features, we think that an effective countermeasure for this category would be URL blacklisting, even if unknown malware is distributed and no anti-virus application can detect it. URL blacklisting for COC is also effective in terms of operating cost and over-blocking. We discuss this in Section 7.2.

7.2. Blacklisting and associated challenges

Blacklisting is widely used as a multilayer defense mechanism in modern Internet security techniques. Microsoft provides the SmartScreen URL Filter [17] to protect users from the sites that are reported to host phishing attacks or distribute malicious software. McAfee provides SiteAdvisor, which adds site rating icons to browser search results [18]. Such alerts can warn against accessing potentially risky sites. Symantec provides Norton Safe Web, which also prevents users from accessing malicious sites [19]. Yandex provides Safe Browsing API SDK [20], through which a library function call can be integrated into a user's project or environment. Google Safe Browsing [21] is installed by default in popular web browsers such as Firefox, Safari, and Chrome. The above products or services are based on blacklisting technology.

The first challenge is to enhance blacklisting coverage. Metcalf et al. compared the contents of 86 Internet blacklists [22] and found that 96.16% to 97.37% of the time domain-name-based blacklists are unique to a list and that 82.46% to 95.24% of the time IP-address-based blacklists are unique to a list. These results show that there is little overlap between each blacklists. Kührer at el. analyzed 15 public blacklists and four anti-virus vendors blacklists [23]. They categorized each blacklist and evaluated which real-world malware domains are covered by the blacklists and found that it was difficult to protect against malware using *Domain Generation Algorithms* [24].

The second challenge is over-blocking and operation cost. Blacklisting increases the effective range of a web service, in order of IP address, FQDN, and URL. Although a higher effective range improves defensive coverage, it also increases the risk of over-blocking. Over-blocking occurs when legitimate site is blocked. For example, over-blocking occurs when we use FQDN blocking for a FQDN site that has both benign URL and malicious URL under the FQDN. Over-blocking is a serious problem, and the best way to prevent it is URL blacklisting for an appropriate period. Note that the IP address blacklist has been installed and operated in traditional firewall. The FQDN blacklist has been installed in a DNS system or `/etc/hosts` file of each PC. The URL blacklist has also been installed in an L7 Firewall, such as Paloalto [25]. When selecting a defense method we should consider both the blocking problem and operation cost. If the lifetime is too short, blacklisting will be ineffective and expensive due to the high costs of delivering a blacklist and installing the defense system. Our observation method and data are applicable to accurate URL blacklisting. Moreover, our analytical results can facilitate the selection of appropriate methods or periods for blacklisting malware download sites.

We next compare our proposed method to the existing countermeasures based on blacklisting technology. Google Safe Browsing blacklists malicious candidate URLs after checking them using a crawler [26]. As there are numerous malicious candidate URLs and limited crawling resources, efficient crawling is necessary. In order to improve the efficiency of blacklisting, the finding of this study can be used to develop a crawling algorithm for determining the crawling interval or period. As opposed to Google Safe Browsing's approach of generating blacklists, our goal in this study is to improve the efficiency of blacklisting by monitoring the status of a blacklisted entries. As the Google Safe Browsing crawling algorithm is not in the public domain, we cannot simply compare technologies in this case. However, we believe that security researchers and engineers engaging blacklisting can apply our findings to their existing systems for blacklisting, e.g., Google Safe Browsing, in order to improve the efficiency of blacklisting operation.

7.3. Observation period adequacy

As described in Section 2.2, we observed URLs for at least 195 days and at most 607 days. Here, we discuss appropriate observation periods. Since the necessary observation period depends on the observation target and purpose, we consider appropriate observation periods for each of the observation target and purpose.

In Section 3.2, we showed that the mean values of *Lifetime* were 203.3, 68.3, and 195.8 days for UNC, ETC, and COC, respectively. Therefore, an observation period of over a half year is required to accurately observe average UNC and COC lifetime. On the other hand, a shorter observation period is sufficient for ETC lifetime.

Some malicious sites resume activities after a long stop period. This is a characteristic not found in regular sites. As discussed in Section 3.2, the *IntervalMax* parameter describes this characteristic. In particular, 1,007 URLs (1,007/43,043 URLs. 2.34%) have *IntervalMax* values that are greater than 100. This means that these URLs revived after over100 days had passed. Long-term observation, i.e., longer than 100 days, is required to reveal such characteristics.

In Section 5, we discussed automatic identification of three categories using machine learning. We showed that an appropriate benchmark observation period to obtain a 5% error rate would be more than three years.

7.4. DNS layer analysis

In our observations, if we obtained an error or no file at all, we assigned the stop status (e.g., Fig. 2, Case_4). We did not distinguish errors in detail, such as HTTP error or DNS errors. In future, if we obtain such detailed error logs, we could analyze the status of malware download sites even further.

7.5. Determination of dynamic re-inspection interval

According to the performance constraints of our observation system, if we received a known malicious file, we assigned the active status to the URL (e.g., Fig. 2, Case_2). If we obtained a known benign file, we assigned the stop status. (e.g., Fig. 2, Case_3). In our definition, we do not consider the case wherein anti-virus applications will change their detection result, malicious or not at a later date. Therefore, it is necessary to determine an appropriate dynamic re-inspection interval.

7.6. Analysis of malware distribution platform

In Section 7.1, we conceived a new type of malware distribution platform. To analyze this new platform in depth, it is necessary to obtain additional data, such as detailed HTTP interactions and HTML content, than we obtained during the observation described in Section 2.1.

7.7. Ethical considerations

We performed web accesses for a long period, and we believe that there is no alternative way to confirm the status of websites. To reduce the amount of unnecessary traffic in our active measurement, we did not conduct consecutive scanning over short periods, i.e., we accessed each website once per day. Note that we downloaded files from websites using a benign procedure, i.e., sending an HTTP GET request, and we did not produce additional intrusive traffic, such as penetration testing and vulnerability scanning.

7.8. Disclosure of observation system

Techniques for thwarting deceptive systems (honeypots) or observation systems are known as *client IP blacklisting*, *emulator detection*, and *fingerprinting* [27–29]. Our observation system changes its IP address once per day in order to circumvent client IP blacklisting. In addition, our system is based on an actual OS and web browser; thus, there is no risk of emulator detection. Our system always has the same fingerprint representing the specific the OS and web browser. In other words, we did not randomize the fingerprint (e.g., changing User-Agent information). Therefore, attackers possibly suspect web clients with the same fingerprint, which sending the same HTTP query for a long period.

8. Related work

A number of studies have analyzed drive-by downloads. Grier et al. [1] investigated the emergence of the exploit-as-a-service model in the drive-by download ecosystem. They showed that many of the most prominent families of malware propagate through drive-by downloads. Provos et al. [3] described the hosting and distribution of drive-by downloads using a very large dataset. They showed that drive-by downloads appear to occur in sudden short-lived spikes. In addition, Shue et al. [30] showed that some ASes are safe havens for malicious activity.

Nappa et al. [4] also investigated the hosting and distribution of drive-by downloads; however, they focused on understanding drive-by download operations. They showed that 10% of exploit servers survived for more than one week, 5% survived for more than two weeks, and some servers survived for up to 2.5 months. Gross et al. [31] presented systems to identify and expose organizations and ISPs that demonstrate persistently malicious behavior. They also showed that drive-by site uptime is longer than that of phishing sites and that 15% of their drive-by dataset survived for more than 60 days. We have demonstrated that compared to exploit servers and malware download sites have much greater longevity.

Zeeuwen et al. [32] focused on changes in malware from malware download centers, and they proposed a scheduling algorithm to obtain new malware. Eshete et al. [33] identified the names of exploit kits using machine learning with 30 HTTP traffic features. As discussed in 4.2, we attempted to do the same using only URLs; however, we found this to be a difficult challenge. Nelms et al. [34] identified the origin of malware download attacks by backtracking the sequence of events. Akiyama et al. [35] showed the activities of attackers after infection using a honeytokens.

Our research differs from previous research in that we focused on analyzing a malware download site's lifetime, revival peculiarity, and malware change-timing model. In particular, the existence of the COC category had not been reveled previously.

9. Conclusion

We have analyzed approximately 43,000 malware download URLs for over 1.5 years and revealed their lifetime and revival peculiarity. We discovered that some malware download sites are long-lived and are revived many times, facts that have not been identified in previous research. We distinguished three categories by focusing on malware variation. Our results show that 10% of the UNC category survives for more than 500 days and that 10% of the COC category was revived more than 15 times. We also analyzed each category's characteristics based on changes in IP address, the number of anti-virus signatures, URL features, and VirusTotal results. We found that each category had different characteristics relative to the attacker's mode of operation and resources. Finally, using our findings, we have discussed effective countermeasures for each category.

References

[1] C. Grier, L. Ballard, J. Caballero, N. Chachra, C.J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, N. Provos, M.Z. Rafique, M.A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Savage, G.M. Voelker, Manufacturing, Compromise, The Emergence of Exploit-as-a-Service, in: Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2012.

[2] 2015 Internet Security Threat Report, Volume 20. Available at http://www.symantec.com/security_response/publications/threatreport.js%p/.

[3] N. Provos, P. Mavrommatis, M.A. Rajab, F. Monrose, All Your iFRAMEs Point to Us, in: Proceedings of the 17th USENIX Security Symposium, 2008.

[4] A. Nappa, M.Z. Rafique, J. Caballero, The MALICIA dataset: identification and analysis of drive-by download operations, Int. J. Inf. Secur. 24 (1) (2015) 15–33.

[5] M. Akiyama, M. Iwamura, Y. Kawakoya, K. Aoki, M. Itoh, Design and Implementation of High Interaction Client Honeypot for Drive-by-Download Attacks, in: IEICE TRANS. COMMUN., 2010.

[6] Trend Micro: ServerProtect for linux. Available at http://www.trendmicro.co.jp/jp/business/products/splx/.

[7] Symantec: Endpoint Protection. Available at https://www.symantec.com/ja/jp/theme.jsp?themeid=endpointsecurity_lineu%p_sep.

[8] McAfee: Endpoint Protection Suite. Available at https://www.mcafee.com/jp/products/endpoint-protection-suite.aspx.

[9] Kaspersky: Kaspersky Security. Available at http://home.kaspersky.co.jp/store/kasperjp/ja_JP/pd/ThemeID.37143200/pr%oductID.5064647600.

[10] Malware Domain List. Available at http://www.malwaredomainlist.com.

[11] urlQuery. Available at https://urlquery.net/.

[12] CLEAN MX realtime database. Available at http://support.clean-mx.de/clean-mx/viruses.php.

[13] Caida: As Ranking. Available at http://as-rank.caida.org.

[14] VirusTotal. Available at https://www.virustotal.com.

[15] Exploring the Blackhole Exploit Kit. Available at https://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit/.

[16] malware-traffic-analysis. Available at http://malware-traffic-analysis.net/.

[17] Microsoft: SmartScreen URL Filter. Available at https://technet.microsoft.com/en-us/library/jj618329(v=ws.11).aspx.

[18] McAfee: SiteAdvisor. Available at http://www.siteadvisor.com/howitworks/index.html.

[19] Symantec: Web of Trust, Norton Safe Web. Available at https://safeweb.norton.com/about?ulang=eng.

[20] Yandex: Safe Browsing. Available at https://tech.yandex.com/safebrowsing/.

[21] Google Safe Browsing. Available at https://developers.google.com/safe-browsing/.

[22] L. Metcalf, J.M. Spring, Blacklist Ecosystem Analysis: Spanning Jan 2012 to Jun 2014, in: Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security, 2015.

[23] M. Kührer, C. Rossow, T. Holz, Paint it Black: Evaluating the Effectiveness of Malware Blacklists, in: Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses (RAID), 2014.

[24] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, D. Dagon, From throw-away traffic to bots: detecting the rise of DGA-based malware, in: Proceedings of the 21st USENIX Security Symposium, 2012.

[25] Palo Alto Networks, Inc. Available at https://www.paloaltonetworks.com/.

[26] T. Gerbet, A. Kumar, C. Lauradoux, A privacy analysis of Google and Yandex safe browsing, in: Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2016.

[27] M.A. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, L. Schmidt, Trends in circumventing web-malware detection, Tech. rep., 2011.

[28] G.D. Maio, A. Kapravelos, Y. Shoshitaishvili, C. Kruegel, G. Vigna, PExy: the other side of exploit kits, in: Proceedings of the 11th International Conference, DIMVA, 2014.

[29] B. Eshete, A. Alhuzali, M. Monshizadeh, P.A. Porras, V.N. Venkatakrishnan, V. Yegneswaran, Ekhunter: A counter-offensive toolkit for exploit kit infiltration, in: NDSS, The Internet Society, 2015.

[30] C. Shue, A.J. Kalafut, M. Gupta, Abnormally malicious autonomous systems and their internet connectivity, IEEE/ACM Trans. Netw. 20 (1) (2012).
[31] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, E. Kirda, Fire: finding rogue networks, in: Proceedings Annual Computer Security Applications Conference, 2009.
[32] K. Zeeuwen, M. Ripeanu, K. Beznosov, Improving malicious URL re-evaluation scheduling through empirical study of the malware download centers, in: Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality(WebQuality), 2011.
[33] B. Eshete, V.N. Venkatakrishnan, Webwinnow: Leveraging exploit kit workflows to detect malicious urls, in: Proceedings of the 4th ACM Conference on Data and Application Security and Privacy(CODASPY), 2014.
[34] T. Nelms, R. Perdisci, M. Antonakakis, M. Ahamad, WebWitness: investigating, categorizing, and mitigating malware download paths, in: Proceedings of the 24th USENIX Security Symposium, 2015.
[35] M. Akiyama, T. Yagi, K. Aoki, T. Hariu, Y. Kadobayashi, Active credential leakage for observing web-based attack cycle, in: Proceedings of Conference on Recent Advances in Intrusion Detection(RAID), 2013.

Yasuyuki Tanaka received the master degree in Institute of Information Security University, Japan in 2015. He is now Ph.D. student in same University. He is security researcher at NTT communications. He has been engaged in research and analyzing new attack code and malware.

Mitsuaki Akiyama received the M.E. degree and Ph.D. degree in Information Science from Nara Institute of Science and Technology, Japan in 2007 and 2013, respectively. Since joining Nippon Telegraph and Telephone Corporation NTT in 2007, he has been engaged in research and development of network security, especially honeypot and malware analysis. He is now with the Cyber Security Project of NTT Secure Platform Laboratories.

Atsuhiro Goto currently serves as Dean and Professor, Graduate School of Information Security, Institute of Information Security (IISEC), Japan. He is a leader of the nationwide graduate schools education program, enPiT-Security, which will bring up promising IT experts with the newest information security professional skills. He also works as Program Director for SIP, Cabinet Office, Government of Japan.