IPSJ Journal Vol. 51 No. 10 1234–1245 (Oct. 2010)

Definition of Affiliation Labels

Regular Paper

Web Tracking Site Detection Based on Temporal Link Analysis and Automatic Blacklist Generation

Akira Yamada,[†1] Masanori Hara[*1] and Yutaka Miyake [†1]

Web tracking sites or Web bugs are potential but serious threats to users' privacy during Web browsing. Web sites and their associated advertising sites surreptitiously gather the profiles of visitors and possibly abuse or improperly expose them, even if visitors are unaware their profiles are being utilized. In order to prevent such sites in a corporate network, most companies employ filters that rely on blacklists, but these lists are insufficient. In this paper, we propose Web tracking sites detection and blacklist generation based on temporal link analysis. Our proposal analyzes traffic at the network gateway so that it can monitor all tracking sites in the administrative network. The proposed algorithm constructs a graph between sites and their visited time in order to characterize each site. Then, the system classifies suspicious sites using machine learning. We confirm that public black lists contain at most 22–70% of the known tracking sites respectively. The machine learning can identify the blacklisted sites with true positive rate, 62–73%, which is more accurate than any single blacklist. Although the learning algorithm falsely identified 15% of unlisted sites, 96% of these are verified to be unknown tracking sites by means of a manual labeling. These unknown tracking sites can serve as good candidates for an entry of a new backlist.

1. Introduction

Web sites can make a wide variety of online services available to users because service providers of those service profit from accociated advertising, such as affiliate programs. The video sharing site called Youtube[⋆1] for example displays advertisements alongside each video clip encouraging users to buy the related items. As another example, the online shopping site, Amazon[⋆2] pays money to those sites that allow users to buy Amazon products. These free online services are therefore supported by advertising or marketing networks with strong links to those free services.

Although users welcome these services, nothing in this world is completely free. When a user accesses a Web site, his/her personal preference or profile information is surreptitiously gathered by the sites. A pair made up of a Web bug, which is a tiny image embedded in a Web page, and its third party cookie is a traditional technique to track users activities over multiple Web sites. If users download a single piece of an invisible image file hosted by a third party site, the Web site can forward the users information to the third party tracking site via a cookie. Therefore, Web tracking site or Web bugs are both latent and serious threats to user privacy when Web browsing[22].

These surveillance activities are more serious in corporate networks, because can potentially disclose corporate secrets. Now that web access has become an important aspect of business, it is difficult to prohibit all Web accesses on the network. However, employees Web access can expose the company interests without anyone being aware. For example, Google Analytics[⋆3] records not only frequency of accesses but also the name of company from which a visitor accesses the site without anyone being aware of it. Furthermore, Nakanohito[⋆4] also records companies' names and geographic location, which are obtained using Whois and DNS information, and makes this information publicly available.

While there are conventional protection mechanisms[1),22)] against such traditional Web bugs, they do not work well against modern tracking techniques. Conventional mechanisms employ the heuristic features of Web bug images so that it is difficult for them to handle other types of hidden objects. In the second generation of Web, called Web 2.0, the tracking techniques are becoming more devious and powerful. Mashup sites, which integrate two or more Web services on a single Web browser potentially violate the same origin policy, which prevents different sites from sharing users' information. Further, such services utilize the

†1 KDDI R&D Laboratories Inc.
∗1 Presently with KDDI Corporation
⋆1 http://www.youtube.com/
⋆2 http://www.amazon.com/
⋆3 http://www.google.com/analytics/
⋆4 http://nakanohito.jp/

new functionalities of browsers, such as JavaScript or Flash, which enable users information to be forwarded to any other site. Therefore, it is difficult to detect and prevent such activities using conventional algorithms.

On the other hand, there are blacklist-based approaches[25], which can filter any type of Web object. However, the lists are not fully adequate because they are generated by Web crawling or through the dedicated work of volunteers. Such crawlers are not good at interpreting JavaScript and Flash, which can embed tracking objects in Web content. In some cases, it is difficult to distinguish whether the content is tracking or not from the content of the text string. In addition, blocking suspicious content seriously affects browsing usability. Therefore, conventional blacklists cannot prevent all tracking sites.

In this paper, we propose Web tracking sites detection and blacklist generation based on temporal link analysis. The system analyzes traffic at the network gateway so that it can monitor every activity in the administrative network. The proposed algorithm constructs a graph between sites and their visited time in order to characterize each site. Since tracking sites inherently linked by many normal sites and are not visited for a long time, the temporal graph can characterize tracking site efficiently. The system then classifies tracking sites using machine-learning algorithms.

We evaluated the accuracy of the system using traffic captured in a corporate network. We first compared 4 public blacklists, and show that they contain at most 15 to 70% of total blacklists. We confirm that the proposed system archive detection rate was 62–73%, which is more accurate than any single blacklist. Although 15% of unlisted sites are falsely classified as suspicious, 96% of the sites are unknown tracking sites. Therefore, by using the results of machine learning-based classification, the system can generate a new blacklist, which contains previously unknown tracking sites.

This paper is organized as follows; Sec. 2 describes tracking site mechanism. In Sec. 3, we describe the system and details of the temporal link analysis algorithm. We evaluate the proposed algorithm in Sec. 4 and discuss the results in Sec. 5. Finally in Sec. 6 we offer our concludes.

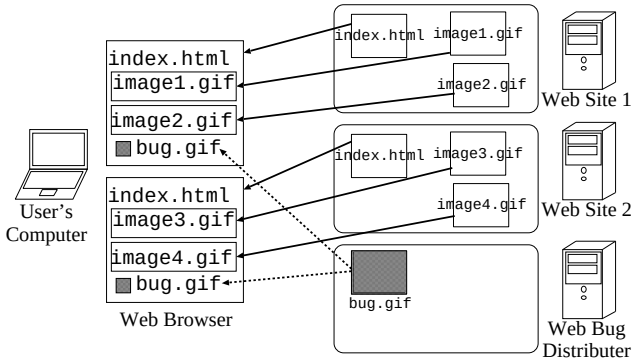

Fig. 1 Mechanism of Traditional Web Bug.

2. Background

2.1 Classic Web Bug

A Web bug is a hidden HTML object, typically a 1x1 pixel or a transparent image, embedded in a Web page designed to track a visitor activity over different Web sites. Fig. 1 illustrates this mechanism. The owners of Web site 1 and 2 are both associated with a Web bug distributor, and put a bug's hyperlink on their pages. The Web bug distributor's site hosts the bug object so that it senses visitors' access whenever they access the embedded pages, even if the distributor does not own site 1 or 2. Many Web sites embed such Web bugs in their pages for the purpose of marketing or analytics,

If the object is an image file, an <IMG> tag is used to allow browsers to download the image automatically. When a user downloads the bug for the first time, the distributor assigns a unique ID to the user, and records it in the user's computer as an HTTP cookie. Therefore, the Web bug distributor can identify users and track them whenever they pass through the embedded pages. The more owners of Web pages associate with the distributor, the more accurately the distributor can track users.

2.2 Modern Tracking Sites

In this paper, we utilize the term "tracking site" rather than "Web bug," because the definition includes that of "Web bug." In the era of Web 2.0, the

environment surrounding Web sites is changing dramatically, and many kinds of services are deployed on Web sites with data in HTML or XML format. These can possess a Web bug-like snooping properties. Considering this situation, we call a Web site that has a tracking function "a tracking site."

A technology introduced in Web 2.0, asynchronous JavaScript and XML (Ajax) enriches users' experience on Web sites, reducing redundant reloads to update the page on the browser. A Web site can manipulate images or text objects without full reloading, so the site is able to pass a user's personal information through the channel. Web browsers follow the same origin policy; however, the policy is permissive toward JSON (JavaScript Object Notation) and Ajax.

In addition, Web 2.0 introduces new types of Web applications and their interfaces, such as Mashup, Web widget and Web API. The new Web application picks some Web services, and mashes them up on a user's Web browser. For example, most blog sites encourage the owners to embed Web widgets at the side or bottom of their page. These new Web applications behave like tracking sites, and possibly possess tracking properties. Therefore, our objective is to detect unwanted tracking functionality and filter it out.

2.3 Related Works

Web advertising services originally employed tracking sites in order to post effective advertisements. As countermeasures against annoying advertisements, there are 2 types of detection or filtering tools in terms of the deployment; browser's extensions and external filtering programs.

Popular Web browsers have a functionality to enhance the behavior of existing features, such as Internet Explorer's Toolbar or Firefox's Extensions. Adblock and its successor Adblock Plus[25] are extensions for Firefox to filter advertisements. Targeting Web bugs in particular, FoxBeacon[26], Bugnosis[1),22)] and a proposal[13] are extensions for FireFox and Internet Explorer. However, the target of these tools is to protect the privacy of personal information rather than a corporate network security.

External filtering programs are installed between the Web browsers and Web servers. For example, configuring an OS's *hosts* file[6),14)], which translates host names into their IP addresses, is a simple but powerful tweak to prevent accessing to unwanted hosts. Alternatively, the Web proxy servers, SquidGuard[28] and DansGuadian[3], Firewall and DNS servers, can serve as filters for tracking of sites.

The detection mechanisms are categorized into blacklist and/or whitelist-based and heuristics and/or learning-based. Early filtering tools[7),24)] based on a blacklist require users to maintain the list to filter advertisements. After that, many volunteers or suppliers start to share or distribute their blacklists. URLBLACKLIST.org[32], which is not free, is a popular blacklist distributor for SquidGuard and DansGuardians, and Easylist[27] is for Adblock Plus. However, detecting all tracking sites and keeping the list up-to-date are problems.

As heuristics-based approaches, the countermeasures in[11),12),20)] introduced size of image files, URL strings, related strings in an HTML file and third party's cookie as the heuristics. As other heuristics, Shin and Karger utilized the table layout on an HTML file[31]. However, these heuristic methods target image-based Web bugs combining a third party's cookie. Therefore, it is difficult to detect presently occurring tracking activities including visible Web widgets that take advantage of Web 2.0 technologies.

Brukner and Voss proposed a Web privacy-preserving suite MozPETs[9], which is installed in a browser, Mozilla. They use a click stream graph which is similar to our link analysis to identify advertisements. However, the tool requires users to install it on their computers in order to observe a user's clicks. In addition, the scheme does not take into the time at which a user visits a site account on the graph.

Most Web sites employ tracking functionalities, which are provided by major companies[16]. Jensen et al. proposed a new Web crawler for Web privacy, and identified many hidden tracking activities through hundred thousands of Web pages[16].

In many cases, if a protection mechanism strictly filters tracking activities, then the usability of browsing is decreased dramatically[19]. Shankar and Karlof proposed a browser-based technique which duplicates a session and examines degradation of usability using another session[29),30)]. However, the method imposes such a large computational load that it is difficult to apply it to an enterprise network.

Link analysis is a useful tool for analyzing Web content, however, they are not

applied to the detection of Web bugs but to extract authoritative or influential Web pages[8),17),18)]. Alternatively, the link analysis algorithm is applied to the social networks analysis[10),23)] on the Web. In order to detect spam Web pages, a variety of metrics are introduced[4),5)] based on a hyperlink graph, however, they do not focus on Web bugs. Because these methods utilize Web crawlers, they analyze the static structure of Web pages.

Combining temporal information with the link analysis algorithm, previous works[2),10)] extracted a trend, such as a hot topic. They utilize a sequence of link edges ordered in time when the edge is generated, rather than visiting time, which is used by our proposal. We explain the proposed combination of visiting time and link analysis in Sec. 3. Luo et al. proposed a framework for temporal link analysis[21)] , but it does not focus on detecting Web bugs.

2.4 Problems of Conventional Tracking Site Detection

Tracking site detection has similar problems to other security tools, such as an intrusion detection system (IDS) or an antivirus (AV) program; a blacklist or/and whitelist-based system needs frequent updating, and heuristics and/or machine learning-based systems detect malicious activities less accurately. In this paper, we focus on the maintenance issue for blacklist-based systems, because most detection software is shifting toward a list-based approach.

A critical problem for heuristic-based systems is new attacks that evade the detection mechanism. In addition, if a heuristic-based algorithm issues a false alert regarding a popular site, then such false alerts are extremely bothersome to users. As a result, even a heuristic-based algorithm adapts a blacklist and whitelist, and becomes hybrid. In actuality, heuristics-based and/or learning-based mechanisms also require maintenance, because the trend of attack gradually changes or some attacks are able to self-mutate.

Currently, most organizations deploy multiple security devices, such as firewalls, URL filtering proxies, and IDSes, which employ rule-based enforcing. A better strategy is to generate a blacklist for the filters. There are conventional blacklists maintained by volunteers[27)] or suppliers[32)], but they cannot comprehend tracking sites because the lists are generated by means of Web crawling or volunteers' contributions. The problems are summarized as follows.

- Blacklist and/or whitelist based detection is more practical, but remains a problem in terms of list maintenance.
- Volunteers' or suppliers' blacklists are insufficient because of Web crawler-based listing.

2.5 Definition of Tracking Sites

We start to define a "tracking site" based on the definition of a Web bug as proposed by Alsaid and Martin[1)] as follows; *Definition (vague). Web bugs are any HTML element that is (1) present at least partially for surveillance purposes and (2) is intended to go unnoticed by users.* Referring to the definition, we define the property of a tracking site as follows;

Property 1 (Surveillance) *An HTML element is present at least partially for surveillance.*

Traditional Web bugs are intended to hide the existence on a Web page, but new types of Web service, such as Mashup or Web widget, are visible but can possibly be utilized for surveillance. For example, an access counter Web widget embedded in a blog can gather and track users' activities. Therefore, we target such unwanted tracking sites, which are loaded automatically. In order to take into account such Web services, we introduce the following properties.

Property 2 (Automatic) *An HTML element is loaded automatically by the user's browser.*

In addition, 1) mentions two other properties as follows; *Property (hostdiff). An element has this property if the host named in its URL is not exactly the same as the host named in the URL of the page containing the element. Property (domaindiff). An element has this property if the host named in its URL is a third party with respect to the URL of the page containing the element.* Because we target tracking hosts rather than domains, we introduce the following modified property.

Property 3 (hostdiff) *An element is hosted by a different host, which distribute elements to multiple independent hosts.*

Considering the above 3 properties, we define a tracking site as follows;

Definition 1 (Tracking Site) A tracking site is a host that distributes any HTML element with (surveillance), (automatic) and (hostdiff) properties. □

Following the definition, traditional 1x1 pixel bugs are definitely categorized as tracking sites. In addition, JavaScript-based analytics tools are also categorized

as such sites. Advertisements embedded in a Web page are also categorized as such sites. In addition, Web widgets that possess a functionality to analyze visitors are also categorized as tracking sites.

To detect tracking sites that have the above 3 properties, we propose temporal link analysis algorithm. The algorithm constructs a temporal graph where sites and links respectively correspond to nodes and edges. The link structure with temporal information characterizes tracking properties.

3. Tracking Sites Detection Based on Temporal Link Analysis

3.1 Overview of Proposed System

Fig. 2 presents an overview of the proposed system, which consists of temporal link analysis, automatic labeling and machine learning-based classification. We assume that a network manager monitors traffic at the network gateway or the proxy server in the administrative network. Since the system does not probe users' computers, we can only observe user actions vaguely and without deal detail, such as moving the mouse or clicking the hyperlink on the browser.

First, the system captures traffic in the network, and characterizes each host based on temporal link analysis. Temporal link analysis consists of visiting time estimation and link analysis algorithms. Second, the system labels each host as a tracking site or not by referring to multiple public blacklists. Finally, the system classifies tracking sites based on machine learning and extracts unknown tracking sites as a new blacklist.

3.2 Temporal Link Analysis

Because of other inheritable properties, users do not stay at the sites for a long time. In addition, Tracking sites are loaded automatically and provide invisible or unwanted information, so users spend less time than on other sites. In order to characterize these properties, we propose a "visiting time" estimation algorithm.

In addition, a tracking site has a property that the site is linked by multiple distinct Web sites. For the purpose of surveillance, the greater the threat is, the more individual sites link a tracking HTML object. Furthermore, users click an element hosted by tracking sites a few times, so that transitions from the tracking site to other sites take a few seconds. To characterize the property, we propose a link graph algorithm, which constructs a link graph between hosts.

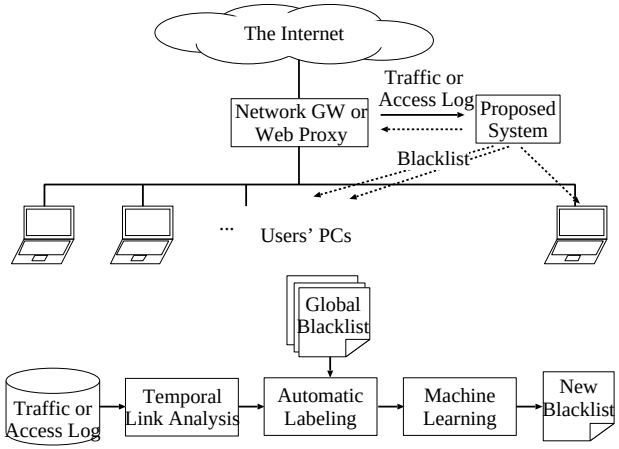

Fig. 2 Overview of Proposed System.

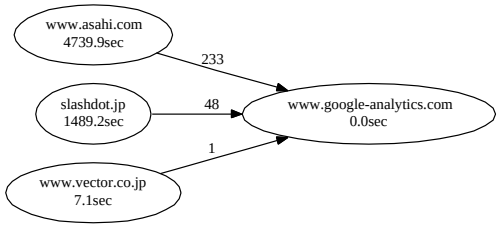

Fig. 3 Concept of Temporal Link Analysis.

Combining the visiting time estimation and link analysis, we generate the proposed temporal graph like Fig. 3. For example, multiple sites link a tracking site www.google-analytics.com, and the tracking site links no other site. Furthermore, users do not spend much time on the tracking site.

3.3 Visiting Time Estimation

In order to estimate a user's visiting time at each host, we extract sequences of URLs clicked by an individual user. Because we target a corporate network, we can identify every user by means of the source IP address. By analyzing a sequence of a user's HTTP requests, we can estimate the user's visiting time at each host.

Due to the limited ability to monitor traffic, it is difficult to collect a user's clicking activities on their computer. We identify a user's clicks using referrer, filename's extension and timing of the requests. When a user accesses a Web site, the browser downloads an HTML file, and then automatically downloads linked objects such as image files. In many cases, the linked objects are non-HTML, so they are not referred by other HTTP requests. On the other hand, the clicked URL is referred by the image files' requests. We therefore extract the previously requested URLs as the candidates of clicked URLs.

We also consider the filename's extension and timing of the request. Since a Cascading Style Sheets (CSS) file can import files from other Web sites, we have to filter such file extensions. In addition, HTTP or HTML's redirections generate referred but not clicked URLs, so we eliminate these URLs using threshold Th. If a URL is identified as having been clicked, then the intervals between the clicks are calculated as the visiting time for each host. We characterize hosts rather than URLs, because some tracking sites alter the URLs at every access.

Algorithm shown in Fig. 4 illustrates how to calculate the visiting time for each host. Let $\{(T_i, Q_i, R_i)|i = 1, 2, \ldots, I\}$ is a sequence of HTTP requests. Q_i and R_i are Request and Referrer URL at time T_i. A users' visiting time B_j for each host $H_j (j = 1, 2, \ldots, J)$ is calculated. A function $index()$ returns the index number of a host in the set of hosts.

3.4 Link Analysis

Using a sequence of HTTP requests, the system constructs a graph, where the vertices and the edges correspond to the hosts and links between the hosts respectively. Since pairs of a request and a referrer are distinguishable, we consider the direction in the graph.

Let $\{(Q_i.host, R_i.host)|i = 1, 2, \ldots, I\}$ is a sequence of request's and referrer's hosts. A request Q_i with a referrer R_i corresponds to an edge from the URL's host $R_i.host$ to $Q_i.host$. Fig. 5 shows the algorithm for constructing a graph $G(V, E)$.

Then, we generate feature vectors, which characterize each host based on the constructed graph. The machine-learning algorithm processes the set of feature vectors at the next step. Eq. (1) describes a feature vector $F()$ of a host H_x.

$$F(H_x) = (f_1(H_x), f_2(H_x), \cdots, f_7(H_x)) \quad (1)$$

Input: $\{(T_i, Q_i, R_i)|i = 1, 2, \ldots, I\}$
// T_i : Time, Q_i : Request URL, R_i : Referrer URL
Output: H, B
// H: A set of hosts $\{H_j|j = 1, 2, \ldots, J\}$
// B: Time visiting j-th host $\{B_j|j = 1, 2, \ldots, J\}$
1: $H \leftarrow \{\}$; $B \leftarrow \{\}$; $J \leftarrow 0$ // Initialize
2: $M \leftarrow \{\}$; // Request URLs pool $\{R_k\}$
3: $T \leftarrow \{\}$; // Time of Request URLs $\{T_k\}$
4: $P \leftarrow ()$; $T_p \leftarrow 0$;
// Previously clicked URL and its time
5: **for all** i **do**
6: **if** $(Q_i.host \notin H)$ **then**
7: $H \leftarrow H \cup Q_i.host$; $J++$; $B_J \leftarrow 0$;
// $Q_i.host$: Q_i's host part
8: **end if**
9: **if** $(R_i.host \notin H)$ **then**
10: $H \leftarrow H \cup R_i.host$; $J++$; $B_J \leftarrow 0$;
// $R_i.host$: R_i's host part
11: **end if**
12: $M \leftarrow M \cup Q_i$; $k = index(Q_i, M)$; $T_k \leftarrow T_i$;
13: **if** $((R_i \in M)$ and
$(T_{index(R_i,M)} - T_p > Th)$ and
(R_i has no specific extension)
// R_i is identified as clicked
14: $M \leftarrow M - R_i$;) **then**
15: **if** $(T_p \neq 0)$ **then**
16: $j \leftarrow index(P.host, H)$;
17: $B_j \leftarrow B_j + T_{index(R_i,M)} - T_p$;
18: **end if**
19: $P \leftarrow R_i, T_p \leftarrow T_{index(R_i,M)}$;
20: **end if**
21: **end for**

Fig. 4 Visiting Time Estimation for a User.

Input: $\{(Q_i.host, R_i.host)|i = 1, 2, \ldots, I\}$
// $Q_i.host$: Request URL Q_i's host part
// $R_i.host$: Referrer URL R_i's host part
// $Q_i.host, R_i.host \in H$
Output: Graph $G(V, E), W(E_\ell)$
// V: A set of vertices $\{V_m\}, V_m \in H$
// E: A set of edge $E = \{E_\ell\}, E_\ell \in H \times H$
// $W(E_\ell)$: Wight of edge E_ℓ

```
1: V ← {}; E ← {}; // Initialize
2: for all i do
3:    if (R_i.host ∉ V) then
4:       V ← V ∪ R_i.host;
5:    end if
6:    if (Q_i.host ∉ V) then
7:       V ← V ∪ Q_i.host;
8:    end if
9:    if (R_i.host, Q_i.host ∉ E) then
10:      E ∪ {(R_i.host, Q_i.host)};
11:      W(R_i.host, Q_i.host) ← 1;
12:   else
13:      W(R_i.host, Q_i.host) + +;
14:   end if
15: end for
```

Fig. 5 Link Graph Construction.

Each feature value is shown in Eq. (2). $f_1(H_x)$ and $f_2(H_x)$ calculate the number of requests and referrers for a host H_x. $f_3(H_x)$ and $f_4(H_x)$ calculate the number of requests and referrers with distinct hosts for a host H_x. $f_5(H_x)$ and $f_6(H_x)$ are the number of edges direct to and from a host H_x. $f_7(H_x)$ is the visiting time on a host H_x.

B_x is the visiting time, which is calculated by the algorithm in Fig. 4. E is the set of edges in the link graph construction. $W(H_j, H_x)$ is a function that returns the weight of an edge (H_j, H_x), which is calculated by the algorithm in Fig. 5.

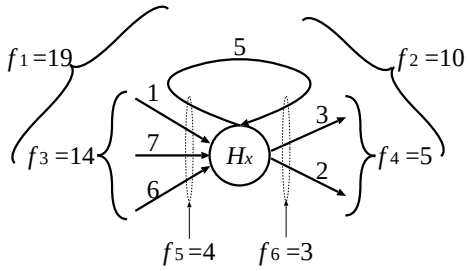

Fig. 6 An Example of a Feature Vector Related to H_x.

We normalize these 7 feature values using the maximum value in the dataset from 0 to 1.

$$\left.\begin{aligned}
f_1(H_x) &= \sum_{\{(H_j,H_x)\in E\}} W(H_j, H_x) \\
f_2(H_x) &= \sum_{\{(H_x,H_j)\in E\}} W(H_x, H_j) \\
f_3(H_x) &= \sum_{\{(H_j,H_x)\in E, H_j\neq H_x\}} W(H_j, H_x) \\
f_4(H_x) &= \sum_{\{(H_x,H_j)\in E, H_j\neq H_x\}} W(H_x, H_j) \\
f_5(H_x) &= \sum_{\{(H_j,H_x)\in E, H_j\neq H_x\}} 1 \\
f_6(H_x) &= \sum_{\{(H_x,H_j)\in E, H_j\neq H_x\}} 1 \\
f_7(H_x) &= B_x
\end{aligned}\right\} \quad (2)$$

Fig. 6 presents a node corresponding to a host H_x in the constructed graph. The number on the each edge means that weight in the graph. For example, the host H_x is requested 19 times and referred 10 times, so the number of requests and referrers are $f_1(H_x) = 19$ and $f_2(H_x) = 10$. Since the host has 5 self-requests, the number of requests and referrers from distinct hosts are $f_3(H_x) = 14$ and $f_4(H_x) = 5$. The number of edges direct to and from are $f_5(H_x) = 4$ and $f_6(H_x) = 3$.

If a host is a tracking site, it gathers many lines from other sites and links to few

other sites. For instance, the more a site is likely to be a tracking site, the higher value f_5 is and the lower value f_6 is. Since the number of incoming/outgoing links depends on the number of total requests and referrers, we have to take into account them as the feature values f_1 and f_2. In some cases, a request leads many other self-link requests in a host. For instance, images embedded in an HTML file lead a lot of requests, which refers the same host. In order to eliminate such cases, we create the feature values f_3 and f_4, which substitute self-links from total number of requests and responses.

We devise the set of feature values experimentally as a relatively optimum solution. We generate several sets of feature values and apply machine learning algorithms. The number of feature values does not affect the performance dramatically, because the value 7 is relatively small in machine learning algorithms. On the contrary, the accuracy is affected, if we reduce one of these feature values. Although it is possible that another combination of values can be a better feature vector, the feature vector proved the best in our experiment.

3.5 Labeling Feature Vectors Based on Public Blacklists

We assign labels to feature vectors as a tracking site or not by utilizing 4 public blacklists. URLBLACKLIST.com[32] distributes one of the largest URL blacklists, which contains advertisement as a category. We also adapt an alternative URL blacklist[15] that targets Japanese Web sites.

A *h*osts file is an alternative way to block advertisement Web sites, and many individuals publish their own *h*osts file on Web sites or blogs. We have chosen MVPS[6]'s *h*osts file and a *h*osts file published on a Japanese blog[14].

The URL blacklists consist of URLs, hosts and expressions, but we use only the hosts for labeling. The reasons are that the feature vectors correspond to the hosts and the *h*osts files contain hosts only. We assign labels to tracking sites if one of 4 lists contains the host as an advertisement. Tab. 1 shows the number of rules for each blacklist.

3.6 Machine Learning Based Classification

We utilize a machine learning suite, Weka 3[33] to classify tracking sites. We selected 5 supervised-learning algorithms, NaiveBayes, BayesNet, J48, KStar, and AdaboostM1. Using the labeled feature vectors, a machine-learning algorithm constructs a prediction model. The constructed model was evaluated with 10-fold cross-validation. It means that we partition randomly the original samples into 10 sub-samples. We use one of 10 samples as training data to validate a model, which is constructed by a machine learning algorithm with the remaining 9 sub-samples of testing data.

Table 1 Blacklists' Rules.

Blacklist	Hosts	URLs	Expressions
URLBL[32]	28,399	994	36
MVPS[6]	15,185	0	0
momo-i[15]	32,258	977	0
lobster[14]	445	0	0

Table 2 Evaluation Datasets.

Dataset	Requests	Hosts
Alexa Top 100 in US	6,727	731
Alexa Top 100 in JP	6,353	534
Corporate Network	133,325,857	156,263
Corporate Network's Top 3,000	–	3000

4. Evaluation

4.1 Datasets

Tab. 2 shows datasets used for the evaluation. We generate Web browsing traffic, displaying popular Web sites with a Web browser Firefox. The datasets contain about 6,000 HTTP requests with 731 and 534 hosts respectively. We collect popular Web sites as seeds, by referring to a Web site Alexa, which summarizes a ranking of popular sites in each country, such as US[⋆1] or Japan[⋆2]

We also use traffic captured in a corporate network over a 3-month period. There are 133 million HTTP requests with 156,000 hosts. Fig. 7 shows the number of HTTP requests for each day. We were unable to obtain HTTP requests for 10 days in July because of a network maintenance. About 200 million HTTP requests are observed on an average weekday. We pick the most accessed 3,000 hosts in the network for machine learning-based prediction.

⋆1 http://www.alexa.com/topsites/countries/US/
⋆2 http://www.alexa.com/topsites/countries/JP/

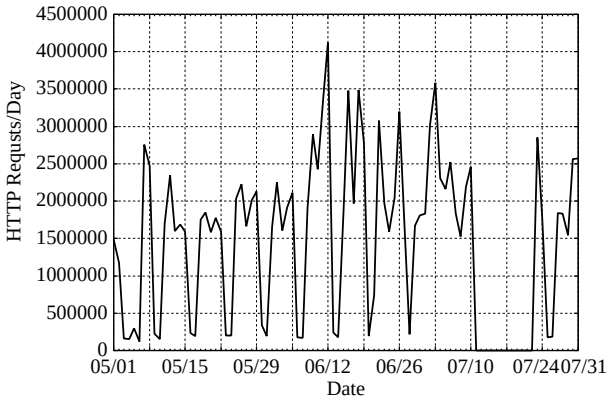

Fig. 7 Daily HTTP Requests in a Corporate Network.

Table 3 Accuracy of Public Blacklists . Each blacklist is compared with the union of 4 blacklists in terms of each dataset.

	Alexa JP	Alexa US	Corp. NW	Corp. NW's Top
URLBL	56.1%	58.7%	63.8%	36.9%
MVPS	61.0%	70.5%	52.0%	41.4%
momo-i	50.4%	65.5%	67.3%	44.5%
lobster	59.3%	15.5%	22.7%	69.2%

4.2 Accuracy of Public Blacklists

In order to evaluate accuracy of publicly available blacklists, we compare the hosts listed by the blacklists using 4 datasets. No single list can achieve 70% of the total blacklists. Tab. 3 shows the accuracy of each dataset and list, referring the union of the 4 blacklists. In the datasets, there are at least several hundreds of tracking sites in the Alexa dataset, and about 3,000 sites in the corporate network dataset.

This result shows that the blacklists depend on the country and size. The blacklists, URLBL, MVPS and momo-i achieve 36% to 70% of tracking sites, because they contains more hosts in the list. On the other hand, a Japanese blacklist, Lobster's detects 16% and 23% of tracking sites in Alexa US. However, Lobster's achieves 59% and 69% in Alexa JP and Corporate Network's Top datasets.

Table 4 Machine Learning-Based Classification (Corp. NW's Top Dataset).

Algorithm	Not Blacklisted		Blacklisted	
	Normal Class	Tracking Class	Normal Class	Tracking Class
/bases/NaiveBayes	2323 (84.9%)	414 (15.1%)	71 (27.0%)	192 (73.0%)
/bases/BayesNet	2325 (84.9%)	412 (15.1%)	98 (37.3%)	165 (62.7%)
/trees/J48	2305 (84.2%)	432 (15.8%)	91 (34.6%)	172 (65.4%)
/lazy/KStar	2326 (85.0%)	411 (15.0%)	90 (34.2%)	173 (65.8%)
/meta/AdaBoostM1	2314 (84.5%)	423 (15.5%)	74 (28.1%)	189 (71.9%)

4.3 Machine Learning Based Classification

Tab. 4 shows the result of classification by the machine learning algorithm. We choose the corporate NW dataset, which contains 263 sites listed and 2,737 not listed by the union of 4 blacklists. Note that there are possibly unlisted but malicious tracking sites in 2,737 sites. The machine learning algorithms can identify 62-73% of listed tracking sites. Since there are 15% of sites that falsely classified as tracking sites, we investigate them in the next section.

4.4 Manual Labeling

We investigate the not listed sites by the blancklists, but classified as tracking sites by means of manual labeling. Tab. 5 shows the 100 sites with manual labels. There are 39 obvious tracking hosts and 26 potential tracking hosts, which possess tracking properties. We cannot distinguish completely whether 31 hosts are tracking sites or not, but they are still suspicious. We suppose that the proposed system archives 96% accuracy to generate a new blacklist.

We categorize mush up sites and blog parts as potential hosts, because such sites actually gather users' information to provide the services. The sites are able to identify visitors, but most users cannot observe the activities. Content Distribution Networks (CDN), such as akamai ⋆1, are labeled as suspicious, since they distribute content based on users' profiles of location. Even if CDNs announce privacy policy, it is possible to leak information accidentally. There are 4 explicit false positives caused by popular Web sites.

Although we label tracking sites based on our policy, each organization has

⋆1 `http://www.akamai.com`

Table 5 Manual Relabeling of Hosts Misclassified by Machine Learning Predication.

Manual Label	Count
Ads/Tracking	39
Potentially Tracking[1]	26
Suspicious[2]	31
False Positive	4
Total	100

1:Mushup and Blog parts potentially posses a tracking function.
2:CDN potentially posses a tracking function

own policy to categorize Web sites. We discuss accuracy of blacklist in Sec. 5.

5. Discussion

5.1 Blacklist's Accuracy

It is hard to identify the purpose of the host in many cases, even if we label manually. We retrieve the content hosted by the server named as the host name, and investigate it. Some hosts redirect the request to the original Web sites so that we can identify the purpose of the Web sites. However, many hosts refuse access without the URL's parameters. We have to guess the purpose of the hosts using information obtained by a Web search engine. However, it is an extremely difficult task to identify the actual purpose of the host, even if manual classification is carried out by a person.

Some Web sites do not distribute the content if the advertisement site is blocked. In such cases, the blocking of tracking sites affects the usability of Web browsing. Therefore, we should assume that many suspicious hosts are not contained in the blacklists. We aggregate 4 blacklists as automatic labeling, but there can be a lot of unlisted hosts.

It is possible that a tracking site has not only a tracking file but also legitimate files. In most actual cases, we cannot encounter significant number of such tracking sites. Popular web sites usually separate the tracking sites from other content for load balancing. For instance, a web site goo.ne.jp has a specific domain name ad.goo.ne.jp rather than goo.ne.jp. If the sites have a specific domain name, the proposed algorithm can identify them.

If such sophisticated tracking sites increase in the future to avoid host-based filters, we can apply our technique to URL-based temporal graph analysis. In this paper, we construct a host-based temporal graph, and we can easily extent our algorithm to URL-based. Regarding one URL as one node in the graph, the algorithm can extract suspicious tracking URLs.

5.2 Blacklist Generation

There are suspicious sites that are incorrectly classified by the machine learning-based prediction, but most of them are true tracking sites. For machine learning, we automatically assign labels based on public blacklists, but these lists are not sufficient. It means that the newly found tracking sites are not contained in any blacklist. Therefore, machine-learning prediction can generate unknown suspicious sites as an update of a blacklist.

Although the accuracy of the generated blacklist is insufficient for a firewall, we believe that it can be applicable for IDSes. We estimate that generated blacklist can achieve 96% of accuracy; there are potential risks when we apply it to firewalls. Since 31% of hosts are not "obvious" but "suspicious" tracking site, the lists possibly generate high false positive. However, the accuracy of generated blacklist is applicable for IDSes, even if it causes many false positives. Since IDSes do not block accesses to the server, error rate is acceptable for observing in the administrative network. In order to observe the administrative network, we believe that more information can contribute to predict possible threats in the network.

In addition, we are considering a confidence metric as a future work. Calculating a confidence value for each host and picking up only high confidential hosts, we can generate a higher accuracy blacklist. When we apply machine learning algorithms, we can obtain confidence value for each label of host. We consider translating these values to a metric of accuracy for the generated blacklist.

6. Conclusion

Web tracking sites or Web bugs are latent and serious threats to users' privacy during Web browsing. In order to protect such sites in a corporate network, most companies employ filters that rely on blacklists, but these lists are insufficient. Because the suppliers collect them mainly based on Web crawling, they cannot comprehend tracking sites. There are too many tracking hosts to collect all of them and to keep them updated. In many cases, it is difficult to distinguish

whether the host is tracking or not.

In order to solve the issue of blacklist updating, we proposed a temporal link analysis algorithm, which constructs a link graph between hosts with visiting time by monitoring network traffic. Because the proposed system is based on passive traffic analysis rather than active crawling, the system can comprehend all Web sites observed in the administrative network. In addition, tracking sites are inherently linked by many distinct sites and visited by users for a very short period, so the algorithm extracts such characteristics using link analysis.

We evaluated 4 public blacklists using traffic captured in a corporate network. Through experimental evaluation, we were able to ascertain that the public blacklists cover only 15–70% of the union of blacklists. This implies that there are many unlisted tracking sites in all over the Web sites. We evaluated the proposed algorithm using captured traffic, and confirmed that the proposed system detects tracking sites with 62–73% of accuracy, which is more than any single blacklist. Although the learning algorithm falsely identified 15% of unlisted sites, 96% of them were confirmed as unknown tracking sites by means of a manual labeling. We believe that the unknown tracking sites can be good candidates for an entry of a new backlist.

References

1) Alsaid, A. and Martin, D.: Detecting Web Bugs With Bugnosis: Privacy Advocacy Through Education, *Privacy Enhancing Technologies*, pp.13–26 (2002).
2) Amitay, E., Carmel, D., Herscovici, M., Lempel, R. and Soffer, A.: Trend Detection Through Temporal Link Analysis, *Journal of the American Society for Information Science and Technology*, Vol.55, No.14, pp.1270–1281 (2004).
3) Barron, D.: DansGuardian – True Web Content Filtering for All, `http://dansguardian.org/`
4) Becchetti, L., Castillo, C., Donato, D., Leonardi, S. and Baeza-Yates, R.: Link-Based Characterization and Detection of Web Spam, *the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pp.1–8 (2006).
5) Benczur, A.A., Csalogany, K. and Sarlos, T.: Link-Based Similarity Search to Fight Web Spam, *the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pp.9–16 (2006).
6) Blocking Unwanted Parasites with a Hosts File: `http://www.mvps.org/winhelp2002/hosts.htm`
7) Boldt, A.: Filtering the Web using WebFilter, `http://math-www.uni-paderborn.de/~axel/NoShit/`
8) Brin, S. and Page, L.: The anatomy of a large-scale hypertextual Web search engine, *Computer Networks and ISDN Systems*, pp.107–117 (1998).
9) Bruckner, L. and Voss, M.: MozPETs - a privacy enhanced Web Browser, *the Third Annual Conference on Privacy, Security and Trust*, pp.21–24 (2005).
10) Buriol, L.S., Castillo, C., Donato, D., Leonardi, S. and Millozzi, S.: Link and Temporal Analysis of Wikigraphs, *the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pp.45–51 (2006).
11) Esfandiari, B. and Nock, R.: Adaptive filtering of advertisements on web pages, *Special interest tracks and posters of the 14th international conference on World Wide Web*, pp.27–31 (2005).
12) Fonseca, F., Pinto, R. and Jr., W.M.: Increasing User's Privacy Control through Flexible Web Bug Detection, *the Third Latin American Web Congress*, pp.205–212 (2005).
13) Gianfrancesco, C., Fiske, A. and Marsh, D.: Extension Based Privacy Protection, Technical report, Worcester Polytechnic Institute (2007).
14) Hatena::Diary id:lobster: no adv, no web, `http://d.hatena.ne.jp/lobster/`
15) Japanese URL Blacklist: `http://jbl.momo-i.org/`
16) Jensen, C., Sarkar, C., Jensen, C. and Potts, C.: Tracking website data-collection and privacy practices with the iWatch web crawler, *the 3rd symposium on Usable privacy and security*, pp.29–40 (2007).
17) Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment, *9th ACM-SIAM Symposium on Discrete Algorithms*, pp.668–677 (1998).
18) Kleinberg, J.M.: Authoritative sources in a hyperlinked environment, *Journal of the ACM (JACM)*, Vol.46, No.5, pp.604–632 (1999).
19) Krishnamurthy, B., Malandrino, D. and Wills, C.E.: Measuring privacy loss and the impact of privacy protection in Web browsing, *the Symposium on Usable Privacy and Security*, pp.52–63 (2007).
20) Kushmerick, N.: Learning to remove Internet advertisements, *the third annual conference on Autonomous Agents*, pp.175–181 (1999).
21) Luo, X., Piret, K. and Guan, Y.: A Service Framework for Temporal Link Analysis with Historical Segments Integration, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp.379–384 (2007).
22) Martin, D., Wu, H. and Alsaid, A.: Hidden surveillance by Web sites: Web bugs in contemporary use, *Commun. ACM*, Vol.46, No.12, pp.258–264 (2003).
23) Mislove, A., Marcon, M., Planck, M., Gummadi, K.P., Druschel, P. and Bhattacharjee, B.: Measurement and Analysis of Online Social Networks, *the 7th ACM SIGCOMM conference on Internet measurement*, pp.29–42 (2007).
24) Muffin World Wide Web Filtering System: `http://muffin.doit.org/`
25) Palant, W.: Adblock Plus: Save your time and traffic, `http://adblockplus.org/en/`

26) Ragkhitwetsagul, C.: FoxBeacon: Web Bug Detector Implementing P3P Compact Policy for Mozilla Firefox, Technical report, Carnegie Mellon University (2007).
27) rick752 &Ares2: Rick 752's EasyList Filters for Adblock Plus, `http://easylist.adblockplus.org/`
28) Shalla Secure Services: SquidGuard, `http://www.squidguard.org/`
29) Shankar, U.: Bridging the Gap between People and Policies in Security and Privacy, PhD Thesis, University of California, Berkley (2006).
30) Shankar, U. and Karlof, C.: Doppelganger: Better browser privacy without the bother, *the 13th ACM conference on Computer and communications security*, pp. 154–167 (2006).
31) Shih, L.K. and Karger, D.R.: Using urls and table layout for web classification tasks, *the 13th international conference on World Wide Web*, pp.193–202 (2004).
32) URLBlacklist.com: `http://urlblacklist.com/`
33) Witten, I.H. and Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*, Morgan Kaufmann (2005).