Privacy Effects of Web Bugs Amplified by Web 2.0*

Jaromir Dobias

TU Dresden, Germany, jaromir.dobias@tu-dresden.de

Abstract. Web bugs are Web-based[1] digital tracking objects enabling third parties to monitor access to the content, in which they are embedded. Web bugs are commonly used by advertisers to monitor web users. The negative impact of web bugs on the privacy of users is known for over a decade. In recent years, Web 2.0 technologies have introduced social aspects into the online media, enhancing the ability of ordinary users to act as the content providers. However, this has also allowed end-users to place web bugs online. This has not only increased the number of potential initiators of monitoring of web surfing behaviour, but also potentially introduced new privacy threats. This paper presents a study on end-user induced web bugs. Our experimental results indicate that, in the light of Web 2.0 technologies, the well-known concept of web bugs leads to new privacy-related problems.

Keywords: Secret Tracking, Social Network, Surveillance, Web 2.0, Web Bugs, Web Privacy

1 Introduction

Web bugs are Web-based digital tracking objects enabling third parties to monitor access to the content, in which they are embedded (e.g., webpages, e-mails or other electronic documents). They have been utilised for several years by Internet marketing or advertising companies for the purpose of tracking and profiling users visiting bugged webpages and analysing their behaviour.

Web bugs are based on a simple mechanism: an HTTP request sent to the tracking server when the tracked content is accessed (see Section 3). This effect can be induced automatically, e.g., by opening webpage or e-mail in a Web browser, e-mail in an e-mail client[2] or document in a text processor, on condition that an external object (e.g., image, script, style sheet, web banner, audio/video stream, etc.) is embedded inside the opened content.

The existence of a significant amount of scientific papers concerning web bugs [1–6] indicates that web bugs are already an old and well-known problem.

* Part of the research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 216483 for the project PrimeLife.

[1] *Web-based* means that they are based on Web technologies.

[2] This behaviour is implicitly disabled by the most of the modern e-mail clients.

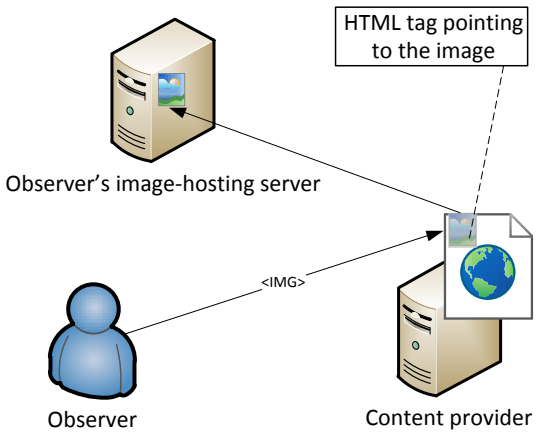

Fig. 1. This figure illustrates observer's deployment of the web bug in the form of an external advertising image. The observer embeds an `IMG` element in the webpage pointing out to the external location of the image under the observer's control.

However, these studies primarily deal with the 'old' World Wide Web, now known as Web 1.0, and hence do not reflect fundamental changes in the nature of the Web environment introduced by Web 2.0 technologies.

In general, web bugs can be utilised for the establishment of a tracking infrastructure across multiple servers, without the need to incorporate tracking functionality to every single server hosting the tracked content. We have discovered that web bugs in Web 2.0 environments (which we will call Web 2.0 bugs), such as online Social Network Sites (SNS), have significantly different properties than traditional web bugs on webpages in the classical Web 1.0 environment.

Web 1.0 bugs are usually tracking devices introduced by, or on request of, an (*observer*), such as a net marketing company or a web analytics company on a web site of a (*content provider*). A practical example might be a content provider intentionally hosting advertising banners loaded from external location under observer's control (see Fig. 1). This allows the observer to monitor access to the content provider's content which embeds the advertising banner. In this model, the content provider provides (dynamic) content, and the observer is only capable of keeping track that the content is visited by end-users. There is relation between the entity initiating the tracking device and the entity hosting content.

This is different with Web 2.0 bugs. If the web bug tracking mechanism is employed into a Web 2.0 environment, the more dynamic creation of content comes into play. Web 2.0 applications are characterized by the fact that end-users are not only consumers, but can also be producers; they can contribute content. Secondly, in Web 2.0 applications end-users can not only contribute content to their own 'domain', but they can also contribute to content provided by others.

Hence, in Web 2.0 environment the technical concept of web bugs can be employed by ordinary users for tracking access to content provided by themselves, as well as provided by others, without these others being aware of the introduction of the tracking mechanism. This shift of monitoring web behaviour by (commercial) entities to end-users 'spying' on each other motivated this research. This paper explores the world of Web 2.0 bugs to study the possibilities of user induced Web 2.0 bugs and explore the privacy effects of those Web 2.0 bugs. The study is set up as a real-world experiments in which we developed a Web 2.0 bug and tested its operation in a real-world Web 2.0 environment.

Outline The remainder of this paper is organized as follows. Section 2 describes existing work related to web bugs. Section 3 describes Web 2.0 bugs and describes the basic settings of the experiments. The results of our experiments are presented in Section 4. Section 5 summarizes the study and presents our conclusions.

2 Related Work

The term "web bug" was introduced by Richard M. Smith. In his 1999 report, *The Web Bug FAQ* [6], he defines web bug as "*. . . a graphics on a Web page or in an Email message that is designed to monitor who is reading the Web page or Email message*". Even though this definition is not incorrect, it omits a wide scale of additional Web-based objects, which can also be used for tracking purposes utilising the same tracking mechanism as web bugs. Additionally, Smith's definition neglects other types of multimedia integrating external Web-based objects, which makes them prone to tracking via web bugs as well (e.g., e-mail clients, word processors, RSS readers, interpreters of Flash content and others). It is therefore necessary to extend the definition of web bugs beyond utilising images for tracking purposes and to include that webpages and e-mails are not the only environments for tracking via web bugs today.

Shortly after releasing his report on web bugs, Smith demonstrated that it is also possible to deploy persistent cookies by web bugs in order to link user requests to their e-mail addresses [5]. After spreading the word about privacy invasive capabilities of web bugs, security researchers, legal scholars, governmental agencies and privacy-aware individuals started to address this problem [2–4].

Trying to raise public awareness, a research team from the University of Denver developed a web bug detection tool called *Bugnosis* [9]. The tool was primarily developed for journalists and policy makers. The authors of Bugnosis hoped, that these two groups might better understand potential threat and privacy impacts of web bugs than the ordinary users and inform the public of the risks and initiate corresponding counteractions.

The idea of warning users against web bugs when browsing websites by Bugnosis was successful at the time, proven by the fact that the tool was installed by more than 100,000 users. The essential problem, however, was that Bugnosis was capable of generating warning messages only, without any defensive steps

actively enhancing user privacy. Another problem was, that the Bugnosis only dealt with external embedded images with specific properties[3]. Last but not least, Bugnosis was developed as an extension plugin for Internet Explorer 5 without support for other browsers or environments. Therefore, Bugnosis is an unsuitable solution for eliminating the privacy threats caused by modern day web bugs.

Even though there are currently some solutions available to disable web bugs [8, 10], none of them deals with the threat of individual users utilising web bugs. As we will see, Web 2.0 bugs introduce novel threats that warrant novel defenses.

3 Web 2.0 Bugs: Web Bugs Employed by Individual End-Users

Web 2.0 bugs are web bugs introduced by users of Web 2.0 applications into the content disclosed through these applications. An example is a user embedding a web bug into a Blogpost on their SNS profile page, or embedding such a web bug into a comment placed on someone else's profile page. This section describes the basic settings of our experiment. It explains technically the principle of tracking via web bugs by individual users. Furthermore, it describes the key aspects of web bug tracking performed by individual users and presents the tracking code utilised in our experiment.

3.1 A Closer Look at Web 2.0 Bugs

An individual user, acting as an (*observer*) can utilise web bugs for tracking all types of media which can carry external objects loaded from the Web. The most common media tracked via web bugs are webpages, while images are the most common external objects used for tracking.

An observer, who wants to track access to a particular webpage, just embeds an external object to the content on the page to be monitored, just like in the case of Web 1.0 bugs. However, in this case the observer is not a company, but an individual end-user.

Next, when an arbitrary visitor opens the webpage containing the embedded object, the visitor's browser automatically generates an HTTP request to retrieve the data of that object from the object-hosting server (see Fig. 2).[4] Each HTTP request incorporates attributes which are to some extent specific to the visitor and which can thus be used for differentiating a particular visitor from others. The HTTP request incorporates:

[3] Bugnosis detected web bugs by evaluating additional properties of images, such as length of the URL, domain of the external image, unique appearance on the webpage, etc [1].

[4] It is assumed that an ordinary modern Web browser is used, which automatically loads images and other external objects embedded on a user requested webpage.

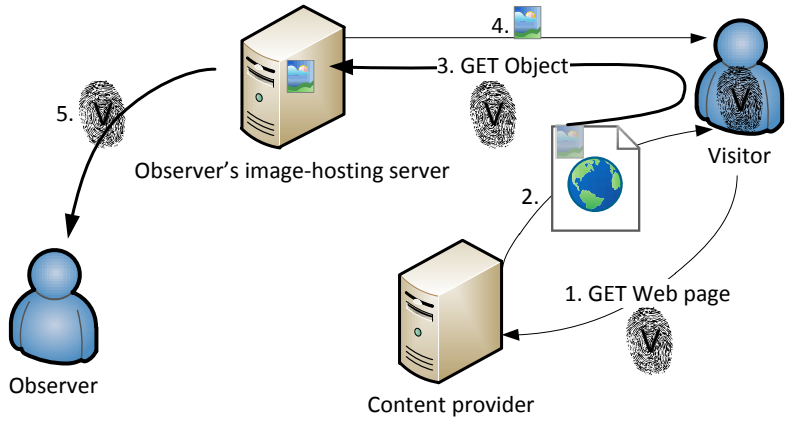

Fig. 2. This figure illustrates tracking via web bugs (image is used in this case). The fingerprint icon symbolises visitor-specific attributes which are contained in each visitor's HTTP request. Paths 3 and 5 depict the side effect caused by the web bug enabling the observer to track the visitor.

- Time when the visitor accessed the tracked webpage;
- IP address of the device which loaded the webpage[5];
- The URL of the tracked webpage[6];
- The URL of the external object (including URL parameters);
- The type of browser accessing the webpage[7];
- Cookie (previously stored in the user's browser by the web bug).

The visitor-specific identifiers contained in the visitor's HTTP requests can be collected on the object-hosting server (step 3 in Figure 2) as soon as he visits the tracked webpage (step 1 in Figure 2). The observer can collect tracked identifiers from each visitor of the tracked webpage on the object-hosting server and secretly monitor access to the tracked webpage (step 5 in Figure 2).

3.2 Key Aspects of Tracking by Individual Users

Nowadays, there are many free public Web-hosting services enabling users to deploy server-side scripts (for instance written in PHP, ASP, or Ruby) on the

[5] Even if the IP address is not specifically associated to the the user's device (user can be behind NAT, VPN, proxy, anonymisation network, etc.), it can reveal some context-dependent information (see Section 4) disclosing information about the visitor.

[6] This information, extracted from the `referer` field of HTTP header can be easily spoofed or removed by the user. In our experiment, we detected however only minor cases in which users intentionally modified this header field (see Section 4).

[7] This information is extracted from the `User Agent` field of the HTTP header. Though it can be easily modified by the user, it provided interesting information as well (see Section 4).

web. Ordinary web users acting as observers can take advantage of these services and use them for hosting the web bugs and collect information about the visitors of the tracked sites. This makes deploying web bugs much easier because it is unnecessary for the observer to run their own dedicated server with public IP address.

The observer might additionally use an anonymisation service to upload the script, deploy the web bugs and afterwards collect the tracked data from visitors of the tracked content. This allows the observer to cover his identity, which is usually not the case when the tracking is done by dedicated Internet marketing or advertising company.

Internet marketing or advertising companies usually host the tracked content directly on devices under their own control.

In contrast, an individual user acting as an observer exploits the fact, that he himself can embed external objects to existing online content and that these objects are loaded from a remote location under his control. Therefore, web bugs provide a covert side-channel enabling individual users to track the content indirectly without the involvement of the content provider. The content providers in this case provide an environment prone to embedding tracking devices. It allows the observer to covertly track other users that interact within the environment created by the content provider.

The motives behind tracking performed by individual users may also differ from those of typical Internet marketing or advertising companies. While such companies are primarily motivated by economical interests, tracking performed by individual users may be driven by motivations such as jealousness (e.g., a jealous husband tracking his wife), sexual motives (e.g., deviants tracking their potential victims), hate (e.g., members of extreme organisations tracking their enemies), etc.

3.3 Background of the Experiment

In our experiment, we used an image object for tracking access to selected webpages. The PHP script (see Listing 1.1) was used for logging the data from the HTTP requests. When an arbitrary user accessed the tracked webpage, his browser generated an HTTP request pointing to the location of the embedded image. As soon as the PHP script was activated by the incoming request, it logged the data corresponding to the request and sent back the data of the image.

The PHP script was titled `index.php` and stored in the directory called `logo.gif`. The advantage of this solution was that the PHP script was activated with a URL in the form of "`www.example.org/logo.gif`", even though `logo.gif` actually activated a PHP script. This way of implementing a web bug makes unnecessary to reconfigure any settings of the object-hosting server, thus allowing the web bug to be hosted on any public web server facilitating PHP scripts.

This solution did not affect the user experience in any way when communicating with the tracked webpage and hence raised no suspicion about the tracking.

In our experiment, web bugs were deployed to selected locations in an experimental social networking platform, the well-known social network *MySpace* and a public profile in a university information system.[8]. Afterwards the tracked data was collected and analysed for the purpose of this experiment.

```
<?php
  $time = date("H:i:s - d.m.y");
  $ip = $_SERVER["REMOTE_ADDR"];
  $ref = $_SERVER["HTTP_REFERER"];
  $req = $_SERVER['REQUEST_URI'];
  $agt = $_SERVER['HTTP_USER_AGENT'];
  $data = "---\n[TIME:] ".$time."\n";
  $data = $data."[IP:] ".$ip."\n";
  $data = $data."[REFERRER:] ".$ref."\n";
  $data = $data."[BUG_LOCATION:] ".$req."\n";
  $data = $data."[AGENT:] ".$agt."]\n";
  $logFile = "logger.log";
  $handler = fopen($logFile, 'a') or die("error");
  fwrite($handler, $data);
  fclose($handler);
  $pic = "./icon.gif";
  header ("Cache-Control: no-cache, must-revalidate");
  header ("Pragma: no-cache");
  header ("Content-Type: image/gif");
  header ("Content-Length: ".filesize($pic));
  readfile ($pic);
?>
```

Listing 1.1. PHP script intercepting data from web bugs.

4 Results

4.1 Web Bugs in the Experimental Social Networking Platform

Several selected forums were tracked by web bugs in the experimental social networking platform[9]. We used images for tracking[10], which were embedded to selected forums via comments. User access to the tracked forums were logged by the PHP script displayed in Listing 1.1.

[8] We are aware that these experiments are not compliant with the EU data protection regulation which prohibits the processing of personal data without a legitimate purpose. We have not informed the users of the various systems that we were collecting personal data, nor have we provided information about the purposes. On the other hand, this is similar to how real web bug exploits would operate.

[9] The Slovak experimental social networking platform `kyberia.eu` served as a basis for the initial phase of our experiments.

[10] The image as a binary object is not necessary for activation of the web bug tracking mechanism. Only the HTML tag `IMG` pointing to the location of tracking script matters.

Users' accesses were logged as long as our comments containing web bugs were displayed to visitors of the tracked forums. The number of displayed comments were adjusted individually by each user of the experimental social networking platform, and therefore, the older our comments with embedded web bug became, the more often user access was not detected via web bug.

The situation was different in forums under our administration. We embedded web bugs directly into the content of the main topic of some forums under our administration. That guaranteed that the tracking was not dependent on the comments of users in that forums which enabled us to detect each user's access to these forums.

Linkage of nicknames Thanks to built-in function of the experimental SNS enabling users to see nickname of a user lastly visiting a particular forum and time of his visit, tracked records from users were linked to their nicknames based on the time correlation (see Listing 1.2 and 1.3).

The arrival time of the user's request was extracted locally on the PHP hosting server by using the PHP `date` command. The extracted time did not always precisely correspond to the last-visited time displayed in the social networking platform (due to communication delay and/or desynchronised clocks). However, this information was sufficient for manual linkage of tracked records to users' nicknames.

```
nickname245   [02-07-2010 - 11:37:48]
nickname475   [01-07-2010 - 09:17:20]
nickname256   [23-06-2010 - 13:46:02]
nickname023   [23-06-2010 - 13:37:00]
...
```

Listing 1.2. Last visited information available for each forum of the experimental social networking platform

```
[TIME:] 13:37:01 - 23.06.10
[IP:] 1.2.3.4
[REFERRER:] http://example.org/forum007
[BUG LOCATION:] /logo.gif/?id=bug-in-forum007
[AGENT:] Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.2.6)
Gecko/20100628 Ubuntu/10.04 (lucid) Firefox/3.6.6]
...
```

Listing 1.3. Tracked record captured by the PHP script linkable to `nickname023`

Visitor-identifying information Once the tracked record was linked to a user's nickname, further information about the user was extracted. This included WHOIS information derived from the user's IP address, information about the user's browser extracted from the `User-Agent` header (providing information identifying the underlying operating system) and information on the URL of the forum tracked by the web bug.

After having assembled this information, we were able to estimate the geographical location of the user and/or (in few cases) the institution from which the user was connecting. That provided us with an interesting picture of the real persons behind the nicknames.

Statistical information As far as each logged record included time information, it gave us a good statistical view on how many users accessed a particular forum in a particular time frame. Based on this data, it was derived which users spent more time on the tracked forum compared to the others, exposing the users' interest in a particular topic discussed in the tracked forum.

Further leaked information Web bugs deployed to an experimental social networking platform allowed us to detect access to tracked forums via Google Cache and Google Translator. In the first case we detected which keywords a user used before visiting the forum (see Listing 1.4[11]).

```
http://webcache.googleusercontent.com/search?q=
cache:FE7_fiAerdEJ:example.org/forum007+XXX+YYY&cd=1&hl=ZZ
```

Listing 1.4. Detected keywords of search request and access via Google Cache

In the second case, we detected the language, which the forum was translated to by the user (see Listing 1.5[12]). According to further information provided by the WHOIS service, the request came from a device residing in the country for which the translation was requested. Based on this information, we assume that someone from the particular foreign country was interested in the topic presented in the tracked forum.

```
http://translate.googleusercontent.com/translate_c?
hl=XX&sl=YY&u=http://example.org/forum007&
prev=/search%3Fq%3DZZZZZ%26hl%3DXX%26sa%3DG
&rurl=translate.google.XX&usg=ALkJrhhjb8ybR1X4
```

Listing 1.5. Detected request on translation of the tracked forum (from `Referer`)

From this experience we learned that web bugs can also reveal very specific actions of the user in case of accessing the tracked content by exploiting the data embedded in HTTP headers (searched keywords in this case).

Note 1. The experimental SNS is primarily targeted for relatively homogeneous group of users, both geographically and linguistically, and therefore request for translation coming from the foreign IP address was considered unusual.

[11] Data was anonymised. Strings "XXX" and "YYY" substitute keywords and "ZZ" substitutes country code.

[12] Data was anonymised. "XX" substitutes code of the targeted language, "YY" substitutes code of source language and "ZZZZZ" substitutes keyword which navigated user to tracked forum.

User profiles We aimed our research on the user profiles as well as a potential resource of data trackable via web bugs. We embedded web bugs to profiles of selected users in the experimental SNS by adding comments with external pictures. Next we collected the same type of data as we did in forums before. In this case however, the data collected by tracking user visits were not related to a specific topic, but rather to specific identity (the profile owner).

Based on the information on users' last visit also incorporated in user profiles, we discovered, that the most frequent visitor of a particular user profile in experimental SNS is the owner of the profile himself. The second interesting finding (specific to the platform) was, that most users visit their own profiles at least once per session. This is probably caused by the design of the platform, which provides access to profile-management functions in the profile interface only.

Most information gained from the profiles of the tracked users originated from owners of the profiles themselves. This involved information on their ISPs, types of browsers, operating systems used and information on how often and when do they access their profiles.

We also detected other users visiting some tracked profiles on a regular basis. We concluded that those users are interested in the particular tracked profiles.

4.2 Web Bugs in *MySpace*

The experimental SNS described in section 4.1 was an easy target for web bugs. It was mainly due to its open and transparent design enabling registered users to gain a lot of information on activities and interests of other users by means of functions built in the SNS. Another reason for that can be geographic, national and linguistic homogeneity within the community of the experimental SNS and relatively small number of its active users (up to 10,000 members). Therefore we changed focus to a more established, and potentially more 'secure' environment, the well-known social network – *MySpace*. We deployed the web bug to a single profile hosted on *MySpace*, whose owner agreed to take part in our research. We intentionally selected this participant because he also had an account in the experimental social networking platform.

Based on the correlation among the tracked data gained from both of his profiles, we detected access to participant's *MySpace* profile from users accessing his profile previously in the experimental social networking platform. Furthermore, we were able to link visitors of the tracked *MySpace* profile to their nicknames from the experimental social networking profile. This deanonymisation of the *MySpace* visitors was possible because we already had sufficient information gained from the experimental social networking platform. Another reason facilitating linking identities was that the set of all potential different identities detected was small enough to find highly probable correlations. The detection of unique identities and linkage would be much more easier, if we would utilise cookie mechanism assigning unique identifier to each new visitor detected. However in our experiments we decided to avoid using cookies because we were interested in the capabilities of simple tracking devices.

4.3 Web Bugs in the Public Profile of the University Information System

In our last experiment, we embedded the web bug to the author's profile in the local university information system, which gave us technical capability to monitor access to it.

We were unable to link detected access to our profile with any of the data previously tracked from MySpace or experimental social networking platform. In other words, none of the users visiting the experimental social network site or the tracked *MySpace* profile visited the author's profile in the University information system (at least, we could not establish plausible links on the basis of the collected data in the experiments). It is important to mention that users of the University information system are identified by their real names while a user-selected nicknames are used for identifying users in the experimental SNS. If a user visiting the author's experimental SNS profile and author's profile in the University information system were detected, it might indicate that it is a user aware of author's real identity and his partial identity related to experimental SNS.

On the other hand, we were able to estimate the probable location of the visitors (at least the country), nationality and operating systems and platforms used by the visitors. We were also able to detect visitors interested in the content of the author's profile by detecting repeated requests from the same source.

In two particular cases, we were able to link the tracked data to requests from identifiable individuals. That was possible because in those cases we had additional information (gained from other channel, i.e. not via web bugs) that an individual, whose identity was known to us, was accessing our profile within a certain time-frame. As far as no further requests from other users visiting our profile within that specific time-frame were detected, we concluded that the intercepted requests originated from the particular identifiable individual. Hence, we were able to link information on the intercepted IP addresses, types of the operating systems and browsers to a natural living person.

4.4 Overall findings

An interesting overall findings of the experiments conducted are, that the information gained by web bugs, relates to (1) the content published in the tracked media, (2) the time span of tracking, (3) the amount of resources tracked by web bugs and (4) the ability of the observer to link tracked requests each other as well as to link them with other user-related information.

Another interesting finding is that in order to find a link among actions performed by a particular user it is not necessary for the intercepted traces (i.e., tracked data in the log file) of the user's actions to be equipped with a globally unique identifier (e.g., session ID contained in the cookie). It is sufficient to have a set of (not necessarily unique) attributes assuring uniqueness within the specific domain (e.g., combination of the IP address together with HTTP `referer` unique within a specific time-frame).

5 Conclusion

Emerging Web 2.0 applications, such as SNSs, contain more and more user-related data and provide enhanced tools supporting interaction among users. That on the other hand also provides a suitable environment for tracking users. In this paper we aimed to experimentally explore potential privacy effects of web bugs amplified by Web 2.0 technologies.

Our small-scale experiments show that the well-known tracking mechanism of web bugs, can be exploited in previously unexplored way – by ordinary users spying other users in Web 2.0 environments. Unlike Internet marketing or advertising companies, who have been known as the most common employers of web bugs, the motivation of users to track other users can go beyond marketing and advertising interests, and thus lead to new potential privacy threats.

We decided to call web bug amplified by Web 2.0 technologies 'Web 2.0 bug' as the term for generalised version of the already known problem. For Web 2.0 bug it is common that it can be exploited by individual users in order to track other users by means of Web 2.0 technologies. Moreover, any Web 2.0 based external object embedded in some content which generates HTTP request automatically when the corresponding content is opened by a visitor can be seen as a potential Web 2.0 bug. Additionally, aside from the webpages or e-mails also other content-providing media must be concerned as an environment suitable for Web 2.0 bug tracking. The text documents, presentation documents, spreadsheets and other types of media allowing external objects to be embedded can also be used for Web 2.0 bug tracking.

The crucial problem is that Web 2.0 bugs exploit a core principle of hypertext – the interconnection of documents – which provides the foundation for Web technologies. Diminishing or eliminating these privacy-impairing effects of the Web 2.0 bugs is therefore difficult.

The most vulnerable applications with respect to Web 2.0 bugs are nowadays SNSs because of their massive concentration of user-related data and user interaction. However, Web 2.0 bugs themselves can not influence functionality of SNSs and therefore are usually underestimated and overlooked by providers of SNSs, which was also experienced in our experiments. On the contrary, as we showed in this paper, Web 2.0 bugs can cause a huge damage to privacy of the users of such sites, especially in case that any user has in fact the ability to perform tracking of other users.

For elimination or mitigation of this problem it is therefore necessary to take into account both points of view – (1) the Web 2.0-based application as well as (2) the user of a the Web 2.0-based application as a potential victim of Web 2.0 bug tracking.

Web 2.0-based applications (especially SNSs as currently the most sensitive environment prone to Web 2.0 bug tracking) should be designed in such a way, that the external content embedded by users does not automatically trigger the Web 2.0 bug tracking mechanism when accessed by visitors. The visitor accessing potentially risky content should be informed about the risk of being tracked by Web 2.0 bugs. Moreover, such risky actions should require visitor's

consent. Some existing SNSs (e.g., *Facebook*) partially solve this problem by creating thumbnail of an external object (e.g., image or video stream), storing this thumbnail on trusted servers and embedding the thumbnail in place of the external object itself. The original external object behind the thumbnail is loaded from its external location on visitor's demand only (e.g., by clicking the play button) which reduces the set of potentially tracked visitors to those wittingly interacting with the remote content.

The user should have a privacy-enhancing tool available which would provide him a user-level protection against Web 2.0 bugs. This is especially important for those cases, when a particular Web 2.0 application itself does not provide protect its users against Web 2.0 bugs. The privacy-enhancing tool should keep track on user's actions and warn the user in case that user's action would cause Web 2.0 bug tracking effect. In such a case the user should be asked whether the external object should be loaded and if his decision should be permanent or temporary concerning the particular object. The user should be able to manage his privacy and specify which domains are trusted for the user. Moreover, the user should also be able to switch among several modes of operation: the (*deny all*) and (*accept all*) modes should enable the user to implicitly deny all traffic potentially having Web 2.0 bug tracking effect (in the *deny all* case) or accept all traffic regardless of the potential Web 2.0 bug tracking effects (in the *accept all* case). The (*accept trusted/ask untrusted*) mode should load all content considered by the user as trusted and ask the user what to do with untrusted content[13]. Last but not least, the (*accept trusted/deny untrusted*) mode should automatically accept all content considered by the user as trusted and deny untrusted content without asking the user.

Currently, the Privacy Dashboard [7] developed under the project PrimeLife seems to be a promising solution dealing with the problem of Web 2.0 bugs. The Privacy Dashboard is an extension plugin for Firefox web browser which helps the user to track, what kind of information websites collect about the user. It enables the user to see e.g. which external websites are used by the currently visited website, if there are some invisible images on the website or if the website enables third parties to track the user across the web. Additionally, it enables the user to block content from external resources and thus proactively disable the Web 2.0 tracking effect.

However, even the current version of Privacy Dashboard[14] does not deal with all aspects of Web 2.0 bugs discussed in this paper. Therefore, building more advanced privacy-enhancing mechanisms solving the problem of Web 2.0 bugs comprehensively remains a challenging task for the future.

References

1. Alsaid, A., Martin, D.: Detecting Web Bugs with Bugnosis: Privacy Advocacy through Education. In: PET'02: Proceedings of the 2nd international conference

[13] *Untrusted content* is content which is not considered to be trusted by the user.

[14] The current version of Privacy Dashboards is 0.8.3 at the time of writing this paper.

on Privacy enhancing technologies. pp. 13–26. Springer-Verlag, Berlin, Heidelberg (2003)
2. Martin, D., Wu, H., Alsaid, A.: Hidden Surveillance by Web Sites: Web Bugs in Contemporary Use. Commun. ACM 46(12), 258–264 (2003)
3. Nichols, S.: Big Brother is Watching: An Update on Web Bugs. Tech. rep., SANS Institute (2001)
4. Office of Inspector General: Use of internet cookies and web bugs on commerce web sites raises privacy and security concerns. Tech. rep., U.S. Department of Commerce (2001)
5. Smith, R.M.: Synchronizing Cookies with Email addresses. `http://www.ftc.gov/bcp/workshops/profiling/comments/rsmith.htm` (1999), online; accessed Jan-2011
6. Smith, R.M.: The Web Bug FAQ. `http://w2.eff.org/Privacy/Marketing/web_bug.html` (1999), online; accessed Jan-2011
7. Primelife privacy dashboard. `http://www.primelife.eu/results/opensource/76-dashboard`, online; accessed Jan-2011
8. Ghostery. `http://www.ghostery.com/`, online; accessed Jan-2011
9. Bugnosis. `http://www.bugnosis.org/`, online, but no longer available; accessed June-2010
10. Web bug detector. `https://addons.mozilla.org/en-US/firefox/addon/9202/`, online (last update April-2009); accessed Jan-2011