Detecting Web Bugs With Bugnosis: Privacy Advocacy Through Education *

Adil Alsaid[1] and David Martin[2]

[1] Saudi Arabian Monetary Agency
The Institute of Banking
aalsaid@cs.du.edu

[2] Boston University
Computer Science Department
dm@cs.bu.edu

Abstract. This article is a case study in the design, implementation, and deployment of the Bugnosis privacy enhancing tool. Downloaded and installed by over 100,000 users to date, Bugnosis contributes to network privacy indirectly — without any technical protection measures such as filtering or anonymization — by raising awareness about Web bugs and arming users with specific information about current Web site practices.

1 Introduction

Bugnosis is an add-on for Internet Explorer that detects Web bugs during Web surfing and alerts the browser operator to their presence. In this article we highlight the most interesting questions and challenges we faced as Bugnosis grew from a summer afternoon exercise into an independent study project and then a full-blown distribution.

We define the Bugnosis notion of Web bugs in section 3.1. Briefly, Web bugs are invisible third party images added to a Web page so that the third party receives notice of the page viewing event. (The name "bug" comes from the term "electronic bug," which is slang for a tiny, hidden microphone. It has nothing to do with a "bug in a program." Web bugs are also variously called Web beacons, pixel tags, and clear GIFs.) A third-party cookie is often associated with the Web bug transaction, which may allow the third party to recognize the user's Web browser uniquely. Web bugs are controversial because not only are they often used to gather information about user behavior without user consent, but they also trick Web browsers into assisting with this surveillance by claiming that an image is required as part of a Web page when in fact the image has no content benefitting the user.

* This work was supported by the Privacy Foundation, the University of Denver, and Boston University. Most of this work was done at the University of Denver Computer Science Department.

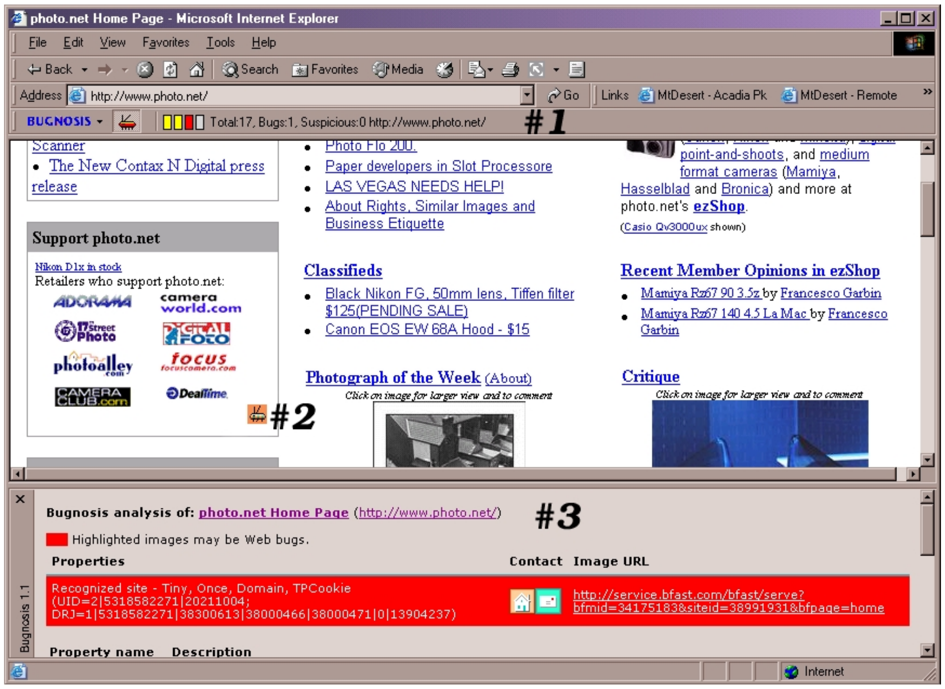


photo.net Home Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.photo.net/
Go Links MtDesert - Acadia Pk MtDesert - Remote
BUGNOSIS
#1
#2
#3
Internet

Total:17, Bugs:1, Suspicious:0 http://www.photo.net/

Bugnosis analysis of: photo.net Home Page (http://www.photo.net/)

🟥 Highlighted images may be Web bugs.

Properties	Contact	Image URL
Recognized site - Tiny, Once, Domain, TPCookie (UID=2|5318582271|20211004; DRJ=1|5318582271|38300613|38000466|38000471|0|13904237)		http://service.bfast.com/bfast/serve?bfmid=34175183&siteid=38991931&bfpage=home

Property name Description

Fig. 1. Screen shot of Bugnosis announcing its discovery of a Web bug. The areas labeled #1, #2, and #3 identify information provided by Bugnosis.

Bugnosis' operation is straightforward. During installation, it attaches itself to Internet Explorer so that it is automatically invoked whenever IE runs. Bugnosis keeps a low profile while the user browses the Web, but it silently examines each viewed Web page for the presence of Web bugs. If it finds a Web bug, it alerts the user with a cute sound ("uh-oh!") and displays information about the Web bug in a separate pane.

Figure 1 shows an Internet Explorer window after Bugnosis has identified a Web bug. In addition to the usual IE paraphernalia, three new areas on the screen are visible and indicated by number. Area #1 is the toolbar Bugnosis control, consisting of a drop-down control menu, a bug-shaped pane visibility toggle switch, a severity meter, and text information about the page. The severity meter progresses from yellow to red. In the original color image, the first three severity bars are lit up, making this a "medium severity" Web bug. The text shows that Bugnosis has analyzed 17 images on the page, found one Web bug, and no merely "suspicious" images. Until a Web bug has been discovered, this toolbar line is the only indication that Bugnosis is present. (Bugnosis 1.1, the currently available version, displays only the visibility toggle switch.)

When Bugnosis discovered the Web bug on the page, it replaced the normally invisible pixels of the bug image with the 18x18 pixel cartoonish image of a bug labeled #2 in the middle area of the screen. On-screen, the bug flashes mildly and gallops along so that the user can spot it easily.

The bottom pane on the screen, labeled #3, shows details about the Web bug itself, such as the URL of the image, the technical properties that lead Bugnosis to consider it a Web bug, and clickable icons. The information here is arranged in a table, but since only one Web bug was discovered, only one row is visible.

2 Philosophy and Strategy

Bugnosis is designed to recognize invisible images served from 3rd-party servers as Web bugs, while excluding those images that seem to be used only for alignment purposes. See sections 3.1 and 3.2 for detailed definitions of Web bugs.

The fascinating thing about Web bugs is that they allow coherent discussion of the well known third party tracking threat: namely, the observation that a cookie-laden third party image provider with wide Internet coverage is capable of gathering very detailed clickstream records about a user population. In the case of Internet advertising firms, this discussion about user tracking is clouded by the fact that the firms' objectives may be totally fulfilled by displaying an advertising image in a Web browser. Although the firms *may* be interested in tracking their user population, the delivery of a third party image is not enough to confirm or deny this suspicion. But Web bugs, by definition, are used only to gather information about users. The question is no longer "*are* they gathering information?" but, by narrowing our interest to Web bug providers, "*why* are they gathering information?" and "what will they do with it?"

2.1 Audience

Our intended audience for Bugnosis is neither whistle-blowers, nor the politically oppressed, nor citizens of states with challenging privacy histories, nor overly scrutinized corporate employees — none of the traditional audience for privacy enhancing tools. Our true audience is journalists and policy makers. This is why: the cookie-based third party tracking threat is just too hard to explain to people. To get from the concept of "a tiny file we store on your computer" all the way to "the ability to track you between Web sites" requires a lot of otherwise irrelevant discussion about Internet topology, HTTP, browser settings, database index keys, and persistence; by then, most people have lost interest in the threat. If journalists and policy makers do not understand the threat, then not much hope remains for the rest of the users.

So our goal was to construct software that educates journalists and policy makers about user tracking. Proceeding under the assumption that most of them are comfortable with a Web browser, and that most probably use Microsoft Internet Explorer for Web browsing and Microsoft Outlook or Outlook Express for e-mail, we could not assume much about their technical sophistication. Therefore, ease of installation and ease of use were absolutely critical to a successful deployment.

2.2 No Web Bug Blocking

An important policy decision we faced was whether Bugnosis should "protect" users by blocking those Web bugs that it identified. Ultimately we decided against it. Although implementation considerations also play into this, we eventually realized that blocking Web bugs had absolutely nothing to do with our education goal. In fact, it would have *distracted* from our goal. If we had added a blocking function, then Bugnosis would have been considered yet another tool for those relatively few people who are so concerned about user tracking that they are willing to invest the time to download, install, and maintain yet another little piece of optional software. But helping a few users block their Web bugs has no impact on the overall tracking regimen, and it only does a little for those users anyway. Basically, adding a blocking feature to Bugnosis would support the opt-out paradigm favored by those behind the tracking systems, precisely by *being* an opt-out mechanism. Without the blocking feature, Bugnosis remains agnostic on this point: it merely informs.

2.3 Low Resistance to Countermeasures

Web bugs are essentially covert channels. Therefore, it's pointless to try too hard to spot them; an adversary that wants to evade Bugnosis can do so easily. Bugnosis simply provides a useful snapshot of current Web site practices.

2.4 Expert Knowledge Database

Although we believe Bugnosis does a good job at identifying Web bugs by their form alone, we recognized early on that an expert knowledge component would provide an important safety valve. Bugnosis therefore incorporates a small database of regular expressions and corresponding dispositions either forcing or preventing matching URLs from being identified as Web bugs. For example, one rule in the database forces the URL http://tnps.trivida.com/tnps/track.fgi to be identified as a Web bug independently of other automated considerations[1]. The database currently contains 23 such "positive" rules, all of which are derived from patterns described by Richard M. Smith [2] in his early investigations of Web bugs. In practice, these rules are seldom decisive; we found that 96% of the Web bugs that Bugnosis identified using these rules in a large sample would have been identified as Web bugs even if the rules had been missing [3].

The database currently contains only 6 "negative" rules preventing the identification of Web bugs. Experience showed us that some certain image forms

[1] TriVida Corp. was acquired by BeFree Inc. in March 2000; this URL appears to be no longer in use, and so the rule is actually obsolete.

served by Yahoo and Akamai that had the formal characteristics of Web bugs were so common that Bugnosis alerted us constantly, even though our inspection of the relevant privacy policies suggested the risk was small. So these URLs are on our negative list. We have to cringe a bit when we manipulate the list, but we also believe that it is important for Bugnosis not to be too alarmist. We have always been more concerned about the possibility of false positives than in overlooking some of the Web bugs.

We included a mechanism to change the database over time. Bugnosis attempts to acquire updated versions of the database every couple of weeks (with the user's permission). This became useful a few weeks after deployment when we discovered an unfortunate interaction between Bugnosis and the WebWasher [4] and Adextinguisher [5] privacy-enhancing tools. These tools block third party images by rewriting their URLs in a way that makes them look like Web bugs to Bugnosis. In the case of WebWasher, the rewritten URL actually appeared to some users to be transmitting information to the WebWasher company, even though a closer look made it clear that this was not true. In early December 2001, we also worked with Google personnel to exclude a set of URLs containing raw IP addresses. Since these URLs were being used only to measure network delay, Google could not change them into a more Bugnosis-friendly form without degrading their measurements. We were able to handle all three of these cases simply by adding negative rules in the database; no other software update was necessary.

The database also contains "neutral" rules that simply associate URL patterns with Web site privacy policies and e-mail contact addresses. This is explained further in section 4.2. Altogether, the Bugnosis database currently contains 45 entries.

3 Web Bugs Defined

3.1 Current Definitions

Definition 1 (vague). *A Web bug is any HTML element that is (1) present at least partially for surveillance purposes, and (2) is intended to go unnoticed by users.*

In order to automatically identify Web bugs, we had to refine this vague definition into something implementable. To get to the first specific Web bug definition, we define some properties that an HTML element may have.

Property 1 (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 2 (domaindiff). An element has this property if the host named in its URL is third party with respect to the URL of the page containing the element. Specifically, this means that their two highest DNS levels (i.e., rightmost two dot-separated components) differ. For example, www.bu.edu is third party to www.du.edu, because bu.edu$\neq$du.edu. (Although this definition is useful in the generic domains, it does not properly capture the notion of third party in some of the country code domains. We return to this issue in section 3.2.)

We are now able to present the two definitions used by other groups of researchers.

Definition 2 (Cy-ID). *A Cy-ID Web bug is an HTML element such that (1) the element is an image, (2) the image is 1x1 pixel large, and (3) the image has the domaindiff property.*

Definition 3 (SS). *An SS Web bug is an HTML element such that (1) the element is automatically loaded, and (2) the element has the hostdiff property.*

The Cy-ID definition describes the Web bugs identified in the Cyveillance study by Murray and Cowart [6] and those elements identified as "invisible objects" in IDcide's Site Analyzer product [7]. This definition captures the notion of "surreptitious" as meaning that the image contains only one pixel. The SS definition, used in the SecuritySpace report by Reinke [8], essentially tags every transaction involving a third party as a surveillance transaction. Since it uses hostdiff rather than domaindiff, it has a very suspicious notion of what a third party is: for instance, a page loaded from www.example.com that references an image from imgs.example.com would be designated a Web bug under this definition.

Stating the Bugnosis definition of Web bugs requires several more properties.

Property 3 (tiny). An image is tiny if it is 7 square pixels or less.

Property 4 (lengthy). Let imageURLs denote the list of all image URLs on the page containing the image in question. An image is lengthy if either (1) imageURLs contains one element and this image's URL contains more than 100 characters, or (2) imageURLs contains more than one element and this image's URL length exceeds $\mu + 0.75\sigma$ characters, where μ and σ are the measured mean and standard deviation of the imageURLs string sizes. An image with this property appears to be communicating something unusual in its URL.

Property 5 (once). An image has this property if its URL appears only once in the list of all image URLs on the page containing the image in question. An image that appears only once on a page is more likely to be a tracking device than an image that appears multiple times.

Property 6 (protocols). An image has the protocols property if its URL contains more than one substring from the set 'http:', 'https:', 'ftp:', 'file:'. For example, an image with the URL http://track.example.com/log/ftp://www.source.com would have this property. This property indicates that its URL may contain some tracking information.

Property 7 (tpcookie). An image has the tpcookie ("third party cookie") property if it has the domaindiff property and the Web browser has a cookie stored for the image's domain.

Property 8 (positive). An image has this property if its URL matches an entry in the "positive" database (see section 2.4).

Property 9 (negative). An image has this property if its URL matches an entry in the "negative" database (see section 2.4).

Given these extra properties, we can now define the Bugnosis notion of a Web bug.

Definition 4 (Bugnosis). *An HTML element is a Bugnosis Web bug if (1) the element is an image, and either (2) it has the positive property, or (3) does not have the negative property but does have the tiny, domaindiff, and at least one additional property other than hostdiff.*

Basically, in order to consider an image a Web bug, Bugnosis requires it to be surreptitious, third-party, and to carry some additional evidence that it is unusual. In practice this means that Bugnosis does not consider tiny images used for spacing purposes as Web bugs, nor does it automatically consider all third party content to be Web bugs.

3.2 Discussion

Although it is certainly possible to obtain perfectly satisfactory surveillance capabilities by using HTML elements other than images, there is no way to distinguish between, say, a JavaScript program that is fetched only in order to trigger a log entry, and one that actually has some "meaningful" content. So Bugnosis only examines images, where it has a hope of telling the difference.

There are only two ways for an image to be considered a Bugnosis Web bug but not a Cy-ID or SS Web bug: (1) if it matches our "positive" database, or (2) if it is just a few pixels too big to be a Cy-ID Web bug. The former case doesn't any introduce error, because our database is built to be consistent with our primary definition 1. The latter case is a possible source of error: we chose to identify images 7 square pixels or smaller to be tiny by just gazing at the screen and determining that images that size were not terribly useful for graphic content, even though they certainly can be visible. Although we haven't observed any problems with our definition — we have indeed seen Bugnosis accurately identify some 1x2 and 2x2 images as Web bugs — 7 square pixels may have been overkill.

We believe the Bugnosis definition of Web bugs is as strong as any other Web bug definition in use. Even so, subsequent Bugnosis versions will make it tighter:

Definition 5 (NG). *An HTML element is a (next generation) Bugnosis Web bug if (1) the element is an image, and either (2) it has the positive property, or (3) does not have the negative property but does have the tiny, strong-domaindiff, tpcookie, and once properties.*

Note that this definition says nothing about the "lengthy" and "protocols" properties. These properties are no longer central to the Web bug identification process; however, once a Web bug has been found, they are used to sort the Web bugs by severity.

We used this tightened definition in our survey of Web bug use [3]. It includes three improvements over the definition used in the currently available Bugnosis release. First, we decided that third party sites that did not use cookies were unlikely to be tracking users individually, since they would have no practical way to distinguish users. (We are not aware of any non-cookie identification techniques in common practice.) On the other hand, although the presence of a third party cookie may indicate individual tracking, it may also be benign, or even privacy enhancing: a cookie that says "TextOnly=true" is a handy way to record a user preference without requiring a full blown registration procedure. Still, since definition 4 also checked for third party cookies, this addition only causes us to identify fewer Web bugs than before.

The second improvement is the requirement of the "once" property, rather than simply taking "once" as some positive evidence of a Web bug. An image whose URL occurs multiple times on the page (and therefore not "once") is clearly not a Web bug: the second URL request would probably not even make it out of the browser's cache, and even if it did, the origin server would end up with two practically identical log entries. No Web designer would naturally repeat a URL as part of a tracking scheme.

Finally, the third change improves our ability to see Web bugs in a wider part of the DNS space. As noted previously, the "domaindiff" notion of "third party" only examined the top two layers of the hosts' DNS names. By this definition, the University of Cambridge Web site www.cam.ac.uk and the University of Oxford site www.ox.ac.uk seem to be in the same domain, which is clearly absurd. We address this with the "strong-domaindiff" property and an auxiliary function called reach[ ] that maps domain name suffixes (com, net, etc.) to domain name levels:

Property 10 (strong-domaindiff). Suppose an element E from the host named H is embedded in a page served from an origin server named O. Let O' be the longest suffix of O such that reach[O'] is defined, and let n=reach[O'], or $n = 3$ if no such O' exists. Then E has the strong-domaindiff property if the top n DNS levels of H and O coincide.

The reach[ ] function is bundled with our expert knowledge database (section 2.4). For example, it specifies reach[uk]=3, reach[com]=2, reach[net]=2, and so on. Thus we are now able to distinguish Cambridge from Oxford, as well as the New York Times from DoubleClick. This was the cleanest solution we could find for the domain test that did not exclude large swaths of the DNS space.

The disadvantage is that by maintaining this mapping ourselves, we run the risk of making mistakes. Fortunately, since the function is distributed with our expert knowledge database, it too can be updated remotely without reinstalling Bugnosis.

We also considered but decided against using the definition of “third party” from RFC 2965 [9], the latest HTTP cookie specification. The issue is that RFC 2965 considers the host “www.example.com” to be third party to the origin server “www.sales.example.com”. While this makes some sense within the DNS zone delegation model, far too many sites use a single authority for everything under their primary domain for this definition to work in Bugnosis. The average Web user would only be confused by the claim that these two sites are substantially different.

RFC 2965 also exhibits some degenerate behavior with short domain names. For example, “images.example.com” is considered third party to the origin server “example.com”, even though “images.example.com” is *not* considered third party to “www.example.com”.

4 Implementation Issues

4.1 Data Transport to and from Bugnosis

We considered using proxy approach in order to discover the images on a page, but our previous experience with proxy-based content understanding [10] was not encouraging. It is very hard to model a Web browser without writing a Web browser. Besides, a central purpose of SSL is to prevent third party snooping, so proxies cannot easily inspect the content of HTTPS pages. (It is possible, though: a trusted proxy can install itself as a root certification authority and use a man-in-the-middle approach to obtain the cleartext [11].)

The Document Object Model (DOM) [12] is well supported by Internet Explorer and provides the hooks we needed. Instead of parsing HTML, Bugnosis asks IE for a DOM representation of the current Web page and then traverses it looking for images and their characteristics. In order to learn about changes in the current document, Bugnosis uses IE’s Browser Help Object [13] functionality to sink relevant events. So Bugnosis sees everything that IE does, even the pages delivered by SSL and those that load images using JavaScript long after the main page has been rendered. In addition, Bugnosis sees the image dimensions as they appear on the page, not as they are recorded in the delivered file. This is appropriate because the “tiny” property has to do with an image’s perceptibility, not its binary content. A 10x10 image file can be made tiny on the screen by changing its size with HTML attributes: `<IMG SRC="http://example.com/10x10.gif" HEIGHT="1" WIDTH="1">`. (It is standard practice among Web designers to specify the height and width attributes with images; without them, the browser’s HTML layout engine has to continually redraw the page as images arrive and it figures out how big they are.)

Having obtained the list of images and their attributes, Bugnosis analyzes all of the images on all of the frames of the Web page and creates an XML representation of the analysis. This XML can be stored for subsequent use (as we did in [3]) or converted immediately for display.

To display its results, Bugnosis creates its own secondary Web browser and embeds it in an Explorer "Comm Band" — the lower pane in the IE window. Bugnosis creates HTML directly (i.e., as text) and writes it into this object to display results. Sticking with HTML is a big advantage here, because it provides a familiar interface with hypertext support that can be easily printed, copied to another window, or e-mailed.

4.2 Making Web Bugs Apparent

Once Bugnosis has identified a Web bug, it sounds an alert and makes its analysis pane visible. In addition, it replaces the image URL in the DOM with a small image of a bug: this immediately causes the bug to become visible in the main IE window. By keeping the visible image small, Bugnosis attempts to leave the page layout mostly intact. Making the bug visible is helpful because Web bugs are often adjacent to other content that may provide a clue as to their purpose. In Figure 1, for instance, the bug is visible next to a list of retailers that support the site being visited; the Web bug probably has something to do with them. Bugnosis also populates the image's ALT tag with summary information about the bug.

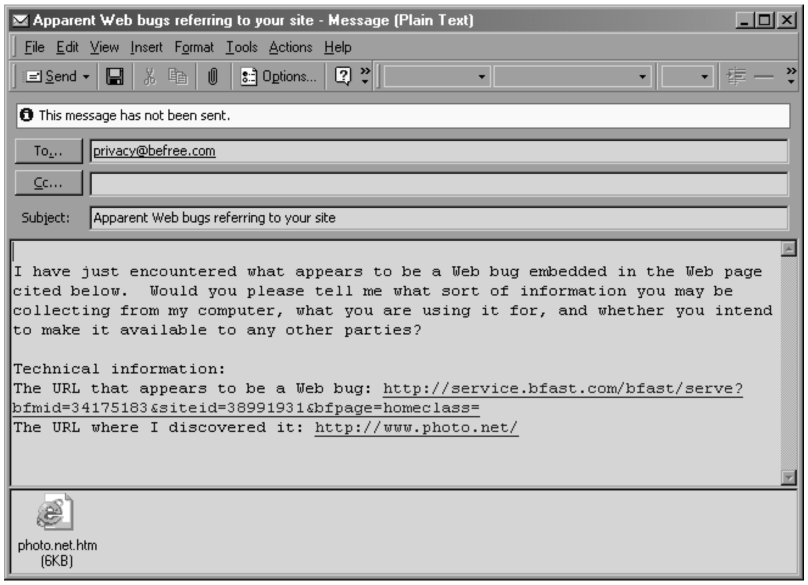
The analysis pane shows the properties supporting Bugnosis' claims about the image; in Figure 1, we see the tiny, once, domaindiff, and tpcookie properties along with the bug's cookie value. The phrase "recognized site" means that the site is in the database of known sites.

The Bugnosis database also associates privacy policy URLs and e-mail addresses with the sites it recognizes. Clicking on the home icon (when present) navigates a new IE window to the privacy policy page for the Web bug provider. Clicking on the e-mail icon (when present) launches the user's e-mail program with a draft message addressed to the Web bug provider, which the user can edit and send if desired.

This e-mail composition (see Figure 2) is by far the most "activist" feature in Bugnosis. Obviously, we felt that Web bug disclosure practices were generally inadequate when we designed this feature. Web bug disclosures may have improved since then [14], but even as late as October 2001, we found that 29% of popular Web sites that contain Web bugs in a U.S. Federal Trade Commission sample say nothing that even hints at the possibility or implications of third party content [3].

4.3 Technical Challenges

Object Soup We were not terribly familiar with COM, ATL, or ActiveX when we began this project, and it turned out to be an excellent, if harrowing, learning opportunity [15, 13]. Bugnosis is packaged as a dynamically linked library (DLL) file that exposes three main COM objects: a tool bar for controlling Bugnosis, the Browser Helper Object for monitoring user-initiated browsing events, and a


Apparent Web bugs referring to your site - Message (Plain Text)
File Edit View Insert Format Tools Actions Help
Send
Options...
This message has not been sent.
To...
Cc...
photo.net.htm (6KB)

To: privacy@befree.com

Subject: Apparent Web bugs referring to your site

```
I have just encountered what appears to be a Web bug embedded in the Web page
cited below.  Would you please tell me what sort of information you may be
collecting from my computer, what you are using it for, and whether you intend
to make it available to any other parties?

Technical information:
The URL that appears to be a Web bug: http://service.bfast.com/bfast/serve?
bfmid=34175183&siteid=38991931&bfpage=homeclass=
The URL where I discovered it: http://www.photo.net/
```

Fig. 2. An e-mail message composed by Bugnosis when the user clicked on the e-mail icon

Comm Band for displaying Bugnosis output. Installation merely adds references to these objects to the system registry. The Bugnosis database is maintained separately in an XML file.

When IE is invoked, it consults the registry and creates these three Bugnosis objects in an apparently unpredictable order, and without any reference to each other. Our first challenge was to make these objects discover each other so they can then share data about the current session. Ultimately we resorted to a global map (shared by every process and thread that loads the Bugnosis DLL) in order to cope with this. During a beta test period we also discovered that some versions of IE for NT maintain a cache that needed to be flushed at installation time; otherwise, IE would not create all three of the objects.

Once the Bugnosis objects have located each other, they create several additional objects: a subordinate Web browser object for displaying Bugnosis output, an XML parser, an auxiliary event sink, and other minor objects. Many of these objects contain references to each other, and this prevents COM's reference counting architecture from freeing inaccessible resources at shutdown time without programmer assistance. Indeed, this was still broken in Bugnosis 1.1; each new IE window leaks some memory until all of the IE threads in that process exit.

Special UI Behavior in the Bugnosis Analysis Pane Not all of the functionality we desired in the Bugnosis output pane was achievable through HTML and scripting alone. In particular, the standard mailto: protocol does not provide a way for a script to include an attachment in an e-mail message, but we felt it was important to include the Bugnosis output when a user sent an e-mail inquiry to a Web bug provider. We also wanted the main IE status line to display appropriate information when the user moved the pointer over links within the Bugnosis analysis window. (The main IE window knows only that the Bugnosis DLL has part of the screen real estate down there — not that Bugnosis is using it to display HTML.) Both issues were solved by attaching an object to the lower pane to sink its events and handling or propagating them appropriately.

We attempted to use IE's "DHTML Behaviors" [13] to automate some of the UI. Behaviors are programmatic content that can be associated with HTML elements in the same way that CSS assigns style information to HTML elements. For instance, we wanted to make the pointer icon and the status line change when the pointer crossed into the e-mail icon so the user would recognize that it is clickable. In addition, we wanted hyperlinks (such as http://service.bfast.com in Figure 1) to invoke a script when clicked, but we wanted them to otherwise look and behave like ordinary links. After inventing a new style for these behaviors and assigning the style to the appropriate objects, we were plagued with unpredictable crashes. Ultimately we stopped using DHTML Behaviors altogether. Instead, one of the DLL objects iterates through the elements and reassigns event handlers whenever the output window changes content.

Finally, we felt it was important to be able to send Bugnosis analyses to computers that did not have Bugnosis installed. Bugnosis therefore removes all of the IE-specific material when preparing its analysis for export (such as printing, saving, copying to the clipboard, etc.).

4.4 Installation and Uninstallation

Since Bugnosis is a collection of COM objects, we thought we would take the ActiveX plunge and use a Web-based installation method. The idea is to provide an installation page that includes an `<OBJECT>` tag. IE will recognize that this tag refers to an ActiveX object that must be fetched and instantiated. Under default settings, this will cause a familiar security dialog to appear asking if the user trusts the software vendor before proceeding. Once instantiated, IE will attempt to display it on the Web page containing the `<OBJECT>` tag. By then, Bugnosis has modified the registry for future IE sessions.

As part of this process it is important to give the user feedback. If anything goes wrong during the ActiveX registration and instantiation, this needs to be explicitly indicated; however, we are restricted to standard scripting techniques for this, since Bugnosis is not yet installed. Our approach is to assume that installation always fails. After the installation page is finished loading, IE switches to a troubleshooting page — unless a certain JavaScript global variable has been set. Meanwhile, whenever Bugnosis senses that the Bugnosis Web site is being visited, it injects a script into the page that sets this variable. So if the Bugnosis installation actually succeeded, it changes the default course of action and instead leads IE to an "installation successful" page.

When the user asks Bugnosis to uninstall itself, it simply disconnects itself from the registry so it will no longer be invoked in later IE sessions. It does not scrape the DLL off the hard drive.

This deployment strategy has not worked as well as we had hoped. Some users expecting the standard download/save/install procedure have complained about unexpected installations even after clicking “install” in response to the explicit security warning. Others feel that the presence of the DLL on the disk after uninstallation was unsafe and requested a way to remove it. But describing how to remove this file is complicated because the precise directory that houses downloaded ActiveX objects is not consistent across all Windows installations. Even worse, the directory has unusual display semantics in Windows Explorer, so it does not show up as expected on the desktop. Even the standard “find file” dialog does not locate it. Luckily, command-line sessions are able to find and manipulate the file. Finally, the installation method just does not work on some systems. We have received many reports of users who saw the “installation failed” page but then later found that Bugnosis was actually working. We clearly have some more work to do here.

4.5 User Community Size

Bugnosis was built by two people with only part-time attention over a period of several months. In the 4.5 months following the initial announcement, over 100,000 users have downloaded and installed Bugnosis. Our Web hosting service quickly upgraded the site to a heavier-duty machine to deal with the initial burst of interest at release time, but we have no good system to deal with the 1,200 e-mail messages we have received so far. Fortunately, much of the e-mail either does not require a response or is covered in the Bugnosis FAQ. Unfortunately, much of the rest simply does not get answered.

5 Plans

5.1 Web Bugs in E-mail

Web bugs are not limited to Web pages. Users equipped with an HTML-enabled mail reader can also be tracked when they read e-mail. When the user views a bugged message, the reader will fetch all of the required images, thereby informing a third party of the reading action, and possibly sending an identifying cookie in the process. Web services exist that automate the construction of Web-bugged e-mails [16, 17].

In order to prepare and send a bugged message, the sender must already have the recipient’s e-mail address. This bit of personal information is generally not available to Web sites that place Web bugs. Since Web bugs in e-mail are usually preserved when the e-mail is forwarded, a tracker may be able to learn who a target’s associates are. And with a little JavaScript, comments added to a message when it is forwarded may even be intercepted by the tracker under some circumstances [18]. For all of these reasons, the practice of bugging e-mails is seen as more intrusive than bugging Web pages. We hope to include an e-mail Web bug detector in a future release.

5.2 Platform for Privacy Preferences Project

Our ability to help the user contact those responsible for the Web bugs is severely limited by our contact database. Without a contact entry, Bugnosis offers no assistance and simply omits the Web page and e-mail icons in its analysis. Furthermore, our database only contains entries about Web bug providers, i.e., the third parties named in the Web bug URL. Thus in the example of Figure 1, Bugnosis provides contact information for "bfast.com", but offers no assistance for contacting "photo.net", even though both sites are necessarily involved in the decision to place the Web bug.

The obvious solution is to use P3P policies [19] to search for both types of contact information. This would allow us to eliminate most of the "neutral" Bugnosis database entries. In addition, we could allow Bugnosis to suppress warnings about Web bugs that have acceptable disclosures. This type of functionality is a high priority for future releases.

Acknowledgments

John Boak, Andy Cervantes, Stephen Keating, and Richard M. Smith also contributed to the Bugnosis project.

References

1. Alsaid, A., Martin, D.: Bugnosis Web bug detector software (2001) http://www.bugnosis.org/.
2. Smith, R.M.: Web bug search page (1999) http://www.computerbytesman.com/privacy/wbfind.htm.
3. Martin, D., Wu, H., Alsaid, A.: Hidden surveillance by Web sites: Web bugs in contemporary use (2001) http://www.cs.bu.edu/fac/dm/pubs/draft-pt.pdf.
4. WebWasher: Webwasher filtering software (2001) http://www.webwasher.com/.
5. AdExtinguisher: Adextinguisher filtering software (2001) http://adext.magenet.net/.
6. Murray, B.H., Cowart, J.J.: Web bugs: A study of the presence and growth rate of Web bugs on the internet. Technical report, Cyveillance, Inc. (2001) http://www.cyveillance.com/.
7. IDCide Inc.: Privacywall site analyzer software (2001) http://www.idcide.com/.
8. Reinke, T.: Web bug report. Technical report, E-Soft Inc. and SecuritySpace (2001) http://www.securityspace.com/s_survey/data/man.200110/webbug.html.
9. Kristol, D., Montulli, L.: HTTP state management mechanism. RFC 2965 (2000)
10. Martin, D., Rubin, A., Rajagopalan, S.: Blocking Java applets at the firewall. In: Proceedings of the 1997 Symposium on Network and Distributed System Security, IEEE (1997) 16–26 See also http://www.cs.bu.edu/techreports.

11. Swiderski, F.: WebProxy auditing and editing tool (2002) http://www.atstake.com/research/tools/index.html#WebProxy.
12. World Wide Web Consortium: Document object model (2001) http://www.w3c.org/DOM/.
13. Microsoft Corp.: Microsoft developer network (2001) http://msdn.microsoft.com/.
14. Network Advertising Initiative: Web bug standards to be developed (2001) http://www.networkadvertising.org/.
15. Roberts, S.: Programming Microsoft Internet Explorer 5. Microsoft Press (1999)
16. Postel Services Inc.: Confirm.to e-mail tracking service (2001) http://www.confirm.to/.
17. ITraceYou.com: Itraceyou e-mail tracking service (2001) http://www.itraceyou.com/.
18. Smith, R.M., Martin, D.M.: E-mail wiretapping (2001) http://www.privacyfoundation.org/privacywatch/report.asp?id=54&action=0.
19. World Wide Web Consortium: Platform for privacy preferences project (2000) http://www.w3c.org/P3P/.