Identifying and Preventing Conditions for Web Privacy Leakage

Craig E. Wills

Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

1 Position Statement

Beyond tracking and proposals to limit it, previous work has observed the *leakage* of private user information to a variety of *third-party aggregators* on the Web via a range of first-party Web sites [3, 5]. These first-party sites include both traditional and mobile Online Social Networks (OSNs) as well as non-OSNs where users register and supply personal information as part of setting up an account.

My position is that we not only need to be concerned with tracking, but also need to identify the conditions under which leakage occurs and work to prevent one or more of these conditions. My work puts me in a position to identify these conditions and point at steps that can be taken by sites and users to prevent them. I focus my attention on leakage to third parties that are present on first-party sites because unlike first-party sites they have the means to observe and *link* the behavior of, and information about, users across multiple first-party sites. I show that while users can take some actions, first-party sites are in the best position to prevent this leakage of private information about their users.

2 Leakage Conditions

Identifying the set of necessary conditions for a problem to occur has been done for other domains. One classic example is for deadlock where [1] identified four conditions that must be present for deadlock to occur and observed that deadlock can be prevented by negating one or more of these necessary conditions. I take a similar approach for the problem of Web privacy leakage where I identify necessary conditions for privacy leakage, examine specific circumstances where these conditions prevail to cause leakage and look at techniques for prevention of leakage by negating one or more of these necessary conditions.

Based upon my own work, I identify three necessary conditions under which I observe the leakage of private information to a third-party aggregator:

1. A user makes information about themselves available to a first-party site. This could be private information such as name or email address, information about their zip code, such as provided through a "store finder" service on a shopping site, or a more-precise latitude/longitude location, such as through a mobile device location.

2. The first-party site receiving this information exposes it in a manner that is visible via HTTP transactions. This exposure is typically through a HTTP request header.

3. A third-party aggregator is present on the first-party site and some amount of user-provided information is made available to the third party as part of a HTTP transaction with the third-party server.

3 Instances of Leakage Conditions

Based on past and current work, I have observed five instances where these three conditions for leakage are realized. While these five instances are not necessarily exhaustive, they do represent the range of leakages that I have observed across both OSN and non-OSN Web sites. In each of the following cases I assume that a user has already provided potentially private information (Condition 1) to a first-party site and show both how the first party exposes this information (Condition 2) in a HTTP transaction to a third-party aggregator (Condition 3). I show representative examples in each case, but intentionally use generic first- and third-party server names to focus on the nature of leakage rather than the specific parties. I have observed a number of instances of all types shown.

3.1 Transmission of User Input via Request-URI

Users provide information about themselves to a first-party site when they edit their user profile or enter terms as part of a search. If this information is transmitted to the first party via the Request-URI then it may be leaked to a third-party in one of two related ways. First it may be leaked to a third-party server via the HTTP `Referer` header if a third-party object is present on the page with the information in the Request-URI. This situation is shown in the following where a zip code is included in the Request-URI by the first party and subsequently leaked by the `Referer` header in a request to `tracker.thirdparty.com`.

```
GET http://tracker.thirdparty.com/params...
Referer: http://www.firstparty.com/...zip=12201...
```

A variant of this leakage occurs when the *next* first-party page a user visits contains third-party JavaScript code that retrieves the referring URL via the JavaScript API and subsequently passes this URL (containing the private information) to the third-party server.

```
GET http://track.thirdparty.com/...
    referer=http://www.firstparty.com/...zip=12201...
Referer: http://www.firstparty.com/nextpage...
```

3.2 Inclusion of Private Information in Page Title

Another example of leakage occurs when first-party sites expose private user information in the title of a Web page, which is then obtained by a third-party script via the JavaScript API. A common example of this type of leakage is when a user's name is put in the title of the user's profile page on a site. This name is subsequently leaked when third-party JavaScript code executes, obtains the page title contents as part of execution and returns it to the third party as part of the Request-URI. Note the example also shows the user's identifier for the site being leaked in the `Referer` header.

```
GET http://tracker.thirdparty.com/...title=John Doe profile...
Referer: http://www.firstparty.com/profile/123456789...
```

3.3 Leakage via First-Party Cookies to Hidden Third-Party

Some sites store private information about the user, such as name or email address, in site-specific first-party cookies. Leakage of this private information occurs when these sites also employ what what is referred to as *hidden third-party* servers where a given server looks like it belongs to a first-party domain, but actually belongs to a third party [4]. An example of this type of leakage is illustrated below where email and full

name are passed to `thirdparty.firstparty.com` because the cookies containing these values are associated with the `firstparty.com` domain and the browser interprets this third-party server as being part of the first-party domain.

```
GET http://thirdparty.firstparty.com/...
Referer: http://www.firstparty.com/...
Cookie:  ...e=jdoe@email.com&f=John&l=Doe...
```

3.4 First-Party Information Used to Populate Third-Party Request-URI

This leakage occurs when information available to the first party is used to populate parameters of a third-party Request-URI. The following example shows such leakage where a user's age, gender and zip code are leaked directly to `tracker.thirdparty.com`. This example demonstrates explicit leakage of first-party information to the third party.

```
GET http://tracker.thirdparty.com/...age=30&gender=M&zip=12201...
Referer: http://www.firstparty.com/...
```

3.5 Information POSTed to Third Party

The final type of leakage was observed in [5] where it was noted that smart phone applications are able to obtain information about a user's device and transmit this information to a third party. The following example shows that a third party is passed the device identifier and latitude/longitude via the API available to the first-party application.

```
POST http://tracker.thirdparty.com/
User-Agent: firstpartyapp/2.2.0 CFNetwork/459

id=IPHONE-UDID,lat=20.00,lon=-70.00
```

4 Leakage Prevention

As noted in [3], third parties receiving private information could filter what is received and not use it. However I believe the right approach is to ensure that third parties do not even receive the information so there is no question on whether or not they are in a position to use it. That leaves two entities—the user and the first-party site—to prevent the leakage of private information by negating one of the three conditions for leakage as defined in Section 2. To illustrate I describe possible actions available to each entity and how each action specifically negates one of the three conditions as well as which leakage instances in Section 3 are prevented.

4.1 User Actions

The simplest approach available to a user is to negate Condition 1 by not providing any private information to a first-party site—a site cannot leak what it does not know. However, creation of an account on a site may be a prerequisite for using the site, such as for an OSN, or the creation of an account may be needed to access valued functionality. In examining a variety of sites on what information is *minimally required* for registration, I found that 95% require an email address while roughly half require some combination of full name, date of birth, zip code and gender.

Given that users do not control first-party site exposure of information, further user-controlled prevention must be done by negating Condition 3. I identify three such user-based actions.

1. One approach that limits leakage via the `Referer` header, as shown in Section 3.1, is to control its use via browser settings. However, Internet Explorer and Safari do provide a setting to control when the `Referer` header is sent and while Firefox and Chrome browsers do provide such a setting, it is disabled by default and requires technical knowledge to enable it [7].

2. Another action available to users for prevention of leakage is to disable JavaScript execution through browser settings or do so selectively via a tool such as NoScript [6]. This action eliminates leakage shown in Sections 3.1 (second example), 3.2 and 3.4 (when information population is done via JavaScript variables). Unfortunately disabling JavaScript execution can negatively affect page quality and cause pages to break [2].

3. Users can use an ad blocker to block all requests to known third-party aggregators. This action is effective when the set of third parties can be identified and negates Condition 3 for examples in Sections 3.1, 3.2 and 3.4. A survey of users on actions taken for privacy protection found that 56% of respondents reported having used ad blockers [8]. However this approach requires that the set of known third parties be maintained and blockage of all hidden third-party servers, as shown in Section 3.3, is difficult.

Other actions available to users regarding blocking cookies or opting out from third-party cookies may inhibit tracking and linking of user records, but do not negate any of the conditions for leakage of information.

4.2 First-Party Actions

Unlike users, first-party sites control what information is exposed in HTTP transactions and can therefore prevent inadvertent leakage by negating Condition 2 via a number of actions.

1. Leakage of the type shown in Section 3.1 can be prevented by not passing the user input via the Request-URI, but by using a HTTP POST method and passing the input as part of the body of the request. With this approach the private information is not exposed in the Request-URI and any third parties will not obtain the information via the `Referer` header.

2. As noted in [3], Facebook uses a variant of this approach by putting a user's identifier after a '#' symbol in the Request-URI. Information after this symbol is not included by browsers in generating the `Referer` header.

3. First-party sites can prevent the "page title" leakage described in Section 3.2 by not putting private information in a Web page title, but rather put it in the contents of the page itself. This approach prevents access to the private information via the JavaScript API.

4. First-party sites can also prevent leakage to hidden third-party servers (Section 3.3) either by not using such servers or alternately changing how cookies are set for a first-party domain. Rather than associate cookies with the domain `firstparty.com`, they should be associated with `www.firstparty.com` so that hidden third parties within the domain (e.g. `thirdparty.firstparty.com`) do not have access to the cookies and therefore cannot obtain their contents.

5. An alternate approach for preventing leakage of the types shown in Sections 3.1 and 3.3 is for first-party sites to hash the private information so that its value is not readable by a third party that may receive the information.

These first-party actions prevent inadvertent leakage of private information by first-party sites, but these actions do not prevent the leakage described in Sections 3.4 and 3.5. The leakage in Section 3.4 shows cooperation by the first-party site to populate the Request-URI so leakage prevention requires the first party to cease such cooperation. The leakage in Section 3.5 is not directly in control of the first-party site and can only be prevented by the first-party application no longer making use of the given third party.

5 Summary

In this position statement I have identified three necessary conditions for private user information made available to a first-party Web site to be leaked to a third-party aggregator. I go on to provide five specific instances of where leakage occurs and show how this leakage can be prevented through a number of actions available to users as well as first-party sites. I believe an understanding of how leakage of private information occurs on the Web is a necessary step in developing technology to help prevent it.

References

[1] E.G. Coffman, Jr., M.J. Elphick, and A. Shoshani. System deadlocks. *ACM Computing Surveys*, 3(2), June 1971.

[2] Balachander Krishnamurthy, Delfina Malandrino, and Craig E. Wills. Measuring privacy loss and the impact of privacy protection in web browsing. In *Proceedings of the Symposium on Usable Privacy and Security*, pages 52–63, Pittsburgh, PA USA, July 2007.

[3] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the Workshop on Online Social Networks in conjunction with ACM SIGCOMM Conference*, August 2009.

[4] Balachander Krishnamurthy and Craig E. Wills. Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the World Wide Web Conference*, pages 541–550, Madrid, Spain, April 2009. ACM.

[5] Balachander Krishnamurthy and Craig E. Wills. Privacy leakage in mobile online social networks. In *Proceedings of the Workshop on Online Social Networks*, June 2010.

[6] Noscript. `http://noscript.net/`.

[7] Christopher Soghoian. Slight paranoia: It is time for the web browser vendors to embrace privacy by default, October 17 2010. `http://paranoia.dubfire.net/2010/10/it-is-time-for-web-browser-vendors-to.html`.

[8] Craig E. Wills and Mihajlo Zeljkovic. A personalized approach to web privacy—awareness, attitudes and actions. *Information Management and Computer Security*, 19(1), 2011.