We are a group of security researchers from the University of Illinois at Chicago (UIC) and are currently investigating different attacks against various browsers' form autofill functionality. We have devised attacks that allow an attacker to surreptitiously obtain sensitive data from users. While Firefox is robust against some of the attacks we have demonstrated across other browsers, we believe that Firefox can still improve its current approach.
Below we provide more details about our findings:

Hiding form elements

The risk of hiding form elements is known. Contrary to Chromium-based browsers that avoid autofilling form elements that have set their CSS *display* property to "none" or their *visibility* to "hidden" or "collapse", Firefox fills all form input elements, even those that are hidden. In the following we present the techniques that we identified for concealing the presence of form fields.

CSS display property. The simplest approach for hiding an element is to set its CSS display property to none. This property completely removes the element and the space it occupies, as if it never existed in the page. Also, this property can be inherited from a parent element.

CSS visibility property. The visibility property specifies whether an element should be visible or not. When this property is set to hidden the element becomes invisible, but its original space and position in the page layout are reserved. It can be also set to collapse, which is treated in the same way as hidden for <input> and <select> elements. Similarly to the display property, the visibility property can be inherited from a parent element.

CSS opacity property. The opacity property specifies the transparency level of the element. When the opacity value is set to 0, the element becomes fully transparent and, thus, invisible to the user. This property is not inherited, but an element cannot be less transparent than its parent.

Covered by overlay. This trick overlays a non-transparent element on top of the element of reference to completely cover it.

Non-effective size. The element is invisible due to its non-effective size (i.e., width or height equal to zero).

Off-screen placement. An attacker can hide an element that has a fixed/absolute position in the page by moving it out of the device's screen area, using the top, bottom, left, and right properties.

Ancestor's overflow. This approach places the element out of the bounds of its ancestor's overflow to make it invisible to the user. This can be implemented in various ways; for example the attacker can set the parent element's height or width equal to zero. Another way, when the ancestor element has an effective size, is to position the element in reference out of the actual

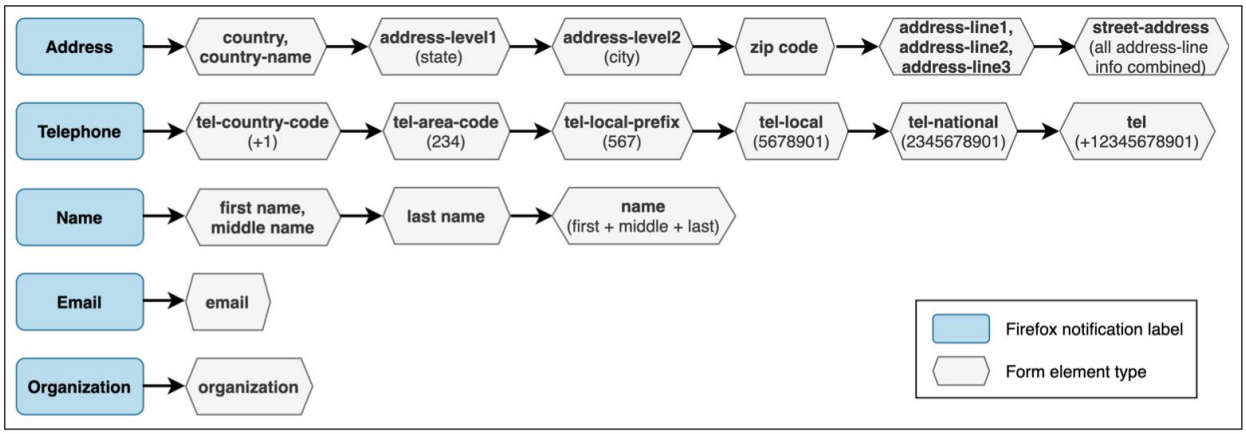
ancestor's bounds and set the ancestor's overflow property to hidden, to disable scrolling functionality for the ancestor element.

Our large-scale study on the Alexa top 100K revealed 5,295 different domains that have at least one form where Firefox will autofill hidden fields. This is often used for deceptive practices (e.g., a form with a visible email input field for a newsletter and hidden fields that obtain the user's full name and street address).

Demo page: https://xlin48.people.uic.edu/project/tricks.html

Autofill Notification Message

When the user starts entering a value in a form element, Firefox shows a notification message informing the user about the types of other information that is autofilled if the autofill functionality is triggered. E.g., "Also autofills address, name, organization". Such a message may help the users identify suspicious behavior in some cases, for instance if a form input field of such a type is hidden. However, the current design can be leveraged to deceive the user.



The problem with this mechanism lies in the granularity of the labels that appear in the message. As can be seen in the figure, multiple types of user information that fall under the same label can have drastically different levels of granularity and, thus, risk if they are leaked. For instance, for all the types of information that are associated with the user's address (i.e., country, state, zip code, city, postal address) the message will simply show the generic label of "Address". This can be exploited by a malicious website to deceive the user into divulging user-identifying information without their consent. An example of such a case would be a domain where a "country" field is visible in the form, while a "street-address" field (also part of the Address category) is hidden. As such, the notification message does not adequately inform the user about what information will actually be provided to the form.
We conducted an analysis and found 650 domains that employ this practice, where the hidden fields obtain more fine-grained user information and visible fields "justify" all the labels in

Firefox's warning message. Thus, we argue that the current design is ineffective as it does not inform users about the true extent of the PII information they are divulging to the website.

Mitigation

For mitigating the aforementioned attacks we suggest:

- Detect form elements that use the techniques described above to hide their presence (i.e., transparent, off-screen), and avoid populating them with autofill. Apply the same restrictions to <select> tag menus.
- Show more fine-grained categories/subcategories for the information that is autofilled in the warning notification.

We can share some code of ours for detecting/preventing such attacks, if you think it will help you fix these issues. Feel free to contact us for more details and information.

Regards,

Xu Lin, PhD Candidate (xlin48@uic.edu)
Panagiotis Ilia, Postdoctoral Researcher (pilia@uic.edu)
Jason Polakis, Assistant Professor (polakis@uic.edu ,
https://www2.cs.uic.edu/~polakis/aboutme/)