November 23, 2015

Abusing CSS Selectors to Perform UI Redressing Attacks

Jovon Itwaru
Information Security Engineer

Introduction

Earlier this year, we received an interesting security advisory from Ruben van Vreeland of BitSensor regarding an issue discovered within our publishing platform. The technique Ruben described is unique and exemplifies the creativity needed to produce high-quality research. We analyzed his report and resolved the vulnerability. While we typically do not talk about bugs that we receive, the lesson learned and the uniqueness of this issue is worth sharing.

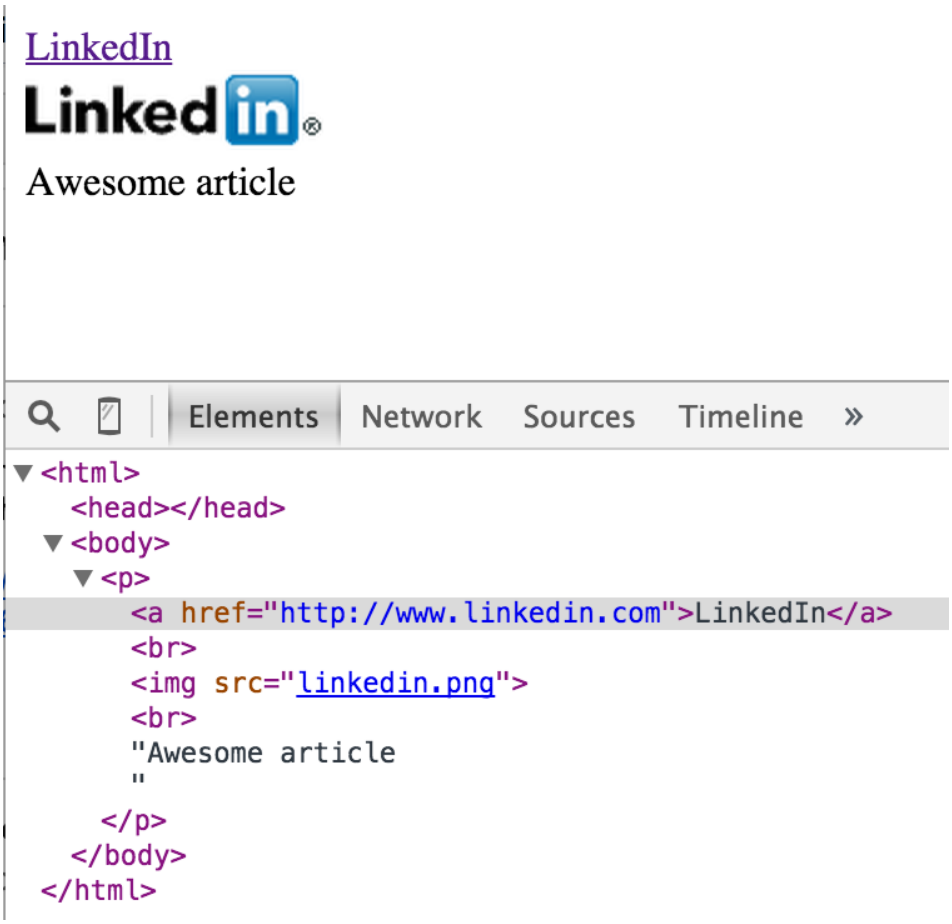
In this blog post, we will describe Ruben's novel attack that allows attackers to use existing CSS and style attributes to trick members into navigating to an attacker-controlled location, leading to potential social engineering and phishing attacks.

Description

As part of our publishing platform, we allow members to customize the look and feel and even share rich media content on their blog articles. This involves styling content with CSS, formatting with a subset of HTML elements, and also sharing audio/video resources. To mitigate certain classes of vulnerabilities such as XSS, a limited set of HTML tags (e.g. `<img>`, `<a>`, `<p>` and `<br>`) and safe attributes are allowed.

Let's dive into a simplified example that illustrates this technique. For instance, to create a blog entry, the following JSON request can be used to generate a new HTML page with an image tag and URL link.

```
json
{"content": "<p><a href=\"http://www.linkedin.com\">LinkedIn</a><img
src=\"linkedin.png\"/></p>"}
```



Resulting HTML page

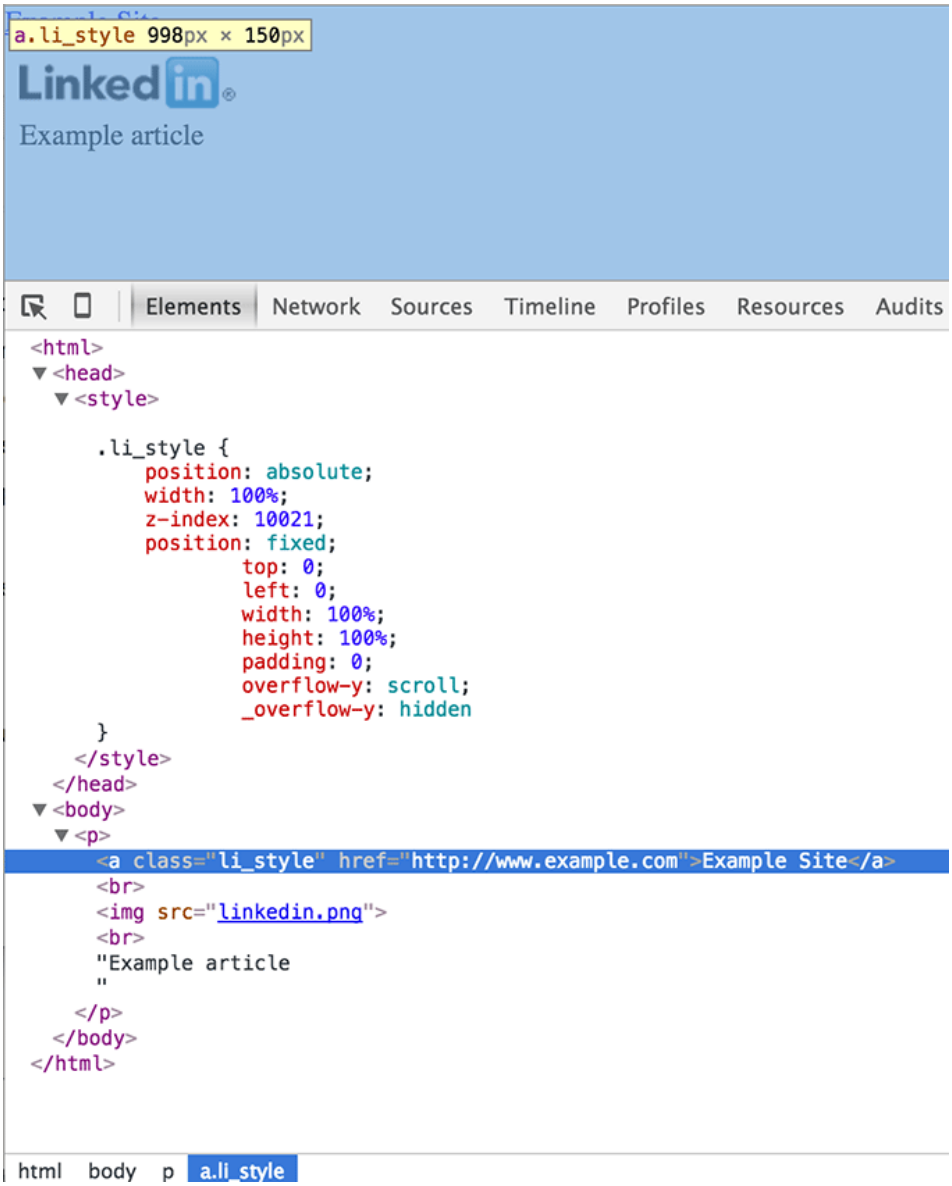
Rigorous input validation is performed on these elements to ensure attackers cannot introduce attribute or event handlers that would be used to construct XSS attacks. In some scenarios, it is possible to introduce benign attributes such as class that will not be flagged by the input validation filter. While this would not be a vulnerability by itself, Ruben realized that it can be used to reference existing CSS hosted on our site. Considering the extent of the platform, we have many CSS classes that are available on our CDNs and consumed by other products. For example, the following CSS styles are applied to the response page that renders blog entries:

```
css
<style>
.li_style {
    position: absolute;
    width: 100%;
    z-index: 10021;
    position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            padding: 0;
            overflow-y: scroll;
            _overflow-y: hidden
}
</style>
```

This type of style is a common way to force an element to stretch the entire height and width of a page. With knowledge of this available CSS style, we can resubmit the request and reference this style:

```
json
{"content": "<p><a class=\"li_style\" href=\"http://www.example.com\">Example
```

```
Site</a><img src=\"image.png\"/></p>"}
```



The `li_style` covers the entire page. This, in turn, allows the page area to become clickable with a link to *http://www.example.com*.

Impact and Recommendation

As illustrated, an attacker can reuse trusted CSS class selectors to perform UI changes that are invisible to members. We believe that this attack is applicable to many sites, as many allow members to create and share rich media content. **This is an interesting technique that uses existing resources to facilitate UI-redressing attacks by chaining together CSS class selectors**, and has similarities to Return Oriented Programming (ROP).

This technique can be used to send members to sites hosting malware or counterfeit sites that attempt to phish members by requesting their usernames and passwords. This is especially successful on social sites that share blogs or articles.

As such, our recommendation is to only accept safe elements and attributes. For example, if the `class` attribute is not allowed, reject any request that contains this. Additionally, whitelist filtering should be applied to CSS class selectors to permit necessary styles.

We would like to thank Ruben for reporting this issue and help keeping our members safe. Thanks to his excellent work and communication with our team, Ruben was invited to join our private bug bounty program, hosted by HackerOne. This is one of many examples of the collaborations we experience with the talented researchers in our program. If you have a bug you would like considered, please submit to security@linkedin.com.
