

# Demo: A Chrome Extension to Prevent the SSLstripping Attack

Daniel Fairweather  
 Computer Science and Engineering  
 New Mexico Tech  
 Socorro, NM 87801, USA  
 dfairwea@nmt.edu

Dongwan Shin  
 Computer Science and Engineering  
 New Mexico Tech  
 Socorro, NM 87801, USA  
 doshin@nmt.edu





## ABSTRACT

SSLstripping is a type of man-in-the-middle attack which has the potential to affect tens of millions of internet users [1]. This attack targets the secure socket layer (SSL) by redirecting users using the HTTP protocol rather than HTTPS. The user, who may not be aware that he/she is using an insecure protocol, submits sensitive data which can then be read by an attacker. Shin and Lopes presented a method for preventing the SSLstripping attack in [1]. This method involves evaluating the form action and page URL in order to determine if a form submission is secure. SSLight is the realization of this method. Implemented as a Google Chrome extension, this tool provides users with a visual security cue which allows users to make better informed decisions when submitting sensitive data. In this demo, we present the latest implementation of SSLight that provides users with an intuitive visual security cue indicating the security status of website forms and a variety of customization options for greater usability.

## 1. INTRODUCTION

SSLight provides users with a visual security cue for website login forms. These visual cues, which resemble traffic lights, appear within the input fields of login forms. Lights come in four colors: green, yellow, red, and gray. Each light indicates the security status of its parent form.

Table 1. Lights and their corresponding indications.

| Light   | Status            | Indication  |
|---|-------------------|---|
|  | Safe to Submit    | Form action uses SSL encryption and belongs to the current host.          |
|  | Submit w/ Caution | Form action uses SSL encryption but redirects away from the current host. |
|  | Unsafe            | Form action does not use SSL encryption.                                  |
|  | Unknown           | The security status has not yet been evaluated.                           |

SSLight allows users to choose how lights should be applied to login forms. These three methods include FormLights, CredentialLights, and PasswordLights. The FormLights method applies lights to input fields whose parent forms contain at least one password field. The CredentialLights method applies lights to username and password fields. Last, the PasswordLights method applies lights to only password fields. These methods provide the

user with varying degrees of visibility, Form Lights covering the most area and Password Lights covering the least.

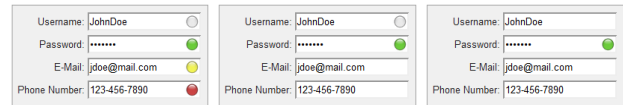


Figure 1. Light application methods. (left to right) FormLights, CredentialLights, PasswordLights

Another feature available in SSLight is the ability to change the images used for lights. By giving the user power to customize the design of SSLight, we can accommodate the visually impaired while simultaneously making the extension more difficult to attack. Users can specify themes or custom images through the extensions option page.

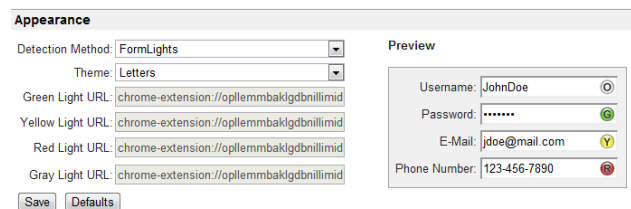


Figure 2. SSLight options page. Preview displays a theme which uses letters superimposed on each light.

## 2. PROCESS

In order to provide the user with real-time security cues, SSLight continuously performs a three-step process which identifies, labels, and analyzes form fields. This process runs on every page the user visits and begins execution after the page has loaded, but before subresources such as images have loaded.

### 2.1 Field Detection

In the field detection phase, SSLight identifies all input fields on the current page which will require a visual security cue. There are three different methods for field detection which the user can select: FormLights, CredentialLights, and PasswordLights. Each of these methods begins by first retrieving all input fields on the current page. The most simple of these methods, PasswordLights, proceeds by identifying which of the fields are password fields. As these fields are found, lights are applied to them. CredentialLights works similarly to PasswordLights in that it first finds password fields. Additionally, a light is added to each password field's previous sibling. FormLights includes one step not included in either of the other methods, password form detection. In this step, we first identify all password fields on the

current page. For each of these password fields, we add their parent form to a list. Once all forms containing a password field have been found, a light is applied to each of the forms' child elements.

## 2.2 Light Application

The light application phase occurs immediately after a field has been identified. First, the computed CSS styles of the input field are gathered. This information is used to apply the light with as little interference with the intended style as possible. The light itself is applied using CSS3 multiple background images. The light image is appended to any other pre-existing background images such that it is superimposed on the other backgrounds. The light is then positioned right and middle within the input field. To prevent text in the field from covering the image, the right padding of the field is adjusted to the width of the light.

## 2.3 Security Analysis

After a light has been placed on an input field, we analyze the security status of both its parent form and the current page. If the current page is using the HTTPS protocol, then SSLstripping is not possible, and therefore, each light should be green. Much more analysis is required when the current page does not use HTTPS. First, we check if the form action is HTTPS. If it is not, a red light is immediately displayed, otherwise, we continue to the next step, certificate checking. The form's action is sent to a remote server where it is then verified via PHP. If the certificate has not expired and is not self-signed, the certificate is considered valid. If the certificate is not valid, a red light is displayed. If the certificate is valid, we compare the form action hostname and the site hostname. If these hostnames are different, we must continue with analysis, otherwise, we display a green light. Last, we check if the site has been whitelisted. If the site has been whitelisted, we display a green light, otherwise, a yellow light is displayed.

```

EVALUATE-FORM()
1  if HTTPS-INITIAL()
2    return green-signal
3  else
4    if formAction ≠ https
5      return red-signal
6    else
7      FURTHER-ANALYSIS()

FURTHER-ANALYSIS()
1  if ¬VERIFY-SSL-CERTIFICATE()
2    return red-signal
3  else
4    if form.act.loc.hostname ≠ doc.loc.hostname
5      if WHITE-LIST(form.act.loc.hostname)
6        return green-signal
7      else
8        return yellow-signal
9  else
10   return green-signal

```

Figure 3. Security assessment algorithm in pseudo code

## 3. DESIGN DECISIONS

### 3.1 Gray Lights

Gray lights were added in order to provide the user with instantaneous feedback. Rather than leave the user guessing as to whether or not a light will be displayed, gray lights let the user know that security information will be available shortly.

## 3.2 Dynamic Analysis

Webpages are subject to dynamic content such as JavaScript events and SSLstripping attacks. Therefore, SSLight performs a routine check in which new forms and fields are identified and changes to current forms are detected. Because of this, SSLight can detect when a light has been removed, or when form actions have been changed.

## 3.3 Background Images

Because SSLight was intended to work on any website, we chose to apply lights as input field background images. By applying lights as background images, minimal modifications to the original site design are made. The result is a highly compatible extension which consistently provides visual cues in the appropriate positions.

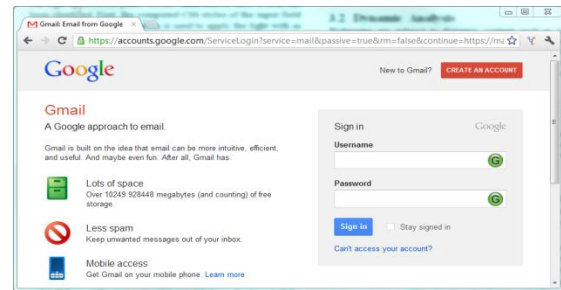


Figure 4. SSLight using the Letters theme

## 4. IMPLEMENTATION

**JavaScript:** The algorithms described above were implemented mostly using JavaScript as this technology is well documented and supported by Google Chrome.

**HTML & CSS:** These markup languages were used to display lights and style the fields. Additionally, all HTML pages such as the options page and the Chrome tooltip were built using HTML.

**PHP:** Certificate checking was accomplished using a PHP script on a remote server. This was done because JavaScript and HTML have no way of verifying SSL Certificates.

## 5. CONCLUSION

In this demo we presented an implementation for preventing the SSLstripping attack by using visual security cues. This implementation provides users with easy to understand and seamless graphics which allow them to make well informed decisions when submitting sensitive data. The current version was tested against more than 100 popular websites with only one style clash.

## 6. ACKNOWLEDGMENTS

This work was partially supported at the Secure Computing Laboratory at New Mexico Tech by the grant from the National Science Foundation (NSF-IIS-0916875).

## REFERENCES

- [1] M. Marlinspike, New tricks for defeating SSL in practice, In Blackhat 2009, Washington DC, 2009.
- [2] D. Shin and R. Lopes 2011. An Empirical Study of Visual Security Cues to Prevent the SSLstripping Attack . In Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC 11), Orlando, Florida, December 5-9, 2011