Weird Tricks to Improve Web Security 10000000%

Yan, TXJS 2015
“Security is sooo tedious.”

- Anonymous web developer
It’s not you, it’s us.
(usually)
We need to make security easy by default.
1. Code injection attacks
2. Man-in-the-Middle attacks

Not covered here: social engineering, phishing, physical access to devices, browser/OS vulnerabilities, TLS vulnerabilities
Problem #1:
User input can cause scripts to be executed on your site that you didn’t intend.
The Mac application for Tweetdeck does not appear to be vulnerable to the XSS. Confirmed in Chrome though.

<script>alert("Yo!");</script>❤

8:55 AM - 11 Jun 2014

Retweets: 36  Favorites: 16
function(e, t) {
    return e ? (t = t || {}), this.updateEntities(e, t), e = this.linkify(e, t), this.emoji(e) : ""
},

_parseTextOnly: function(e, t) {
    for (var i = e.childNodes, r = n.length, s = 0; r > s; s++) i = n[s], 3 === i.nodeType & TD.emoji.test(i.nodeValue) ? t.push(i) : i.hasChildNodes() && this._parseTextOnly(i, t)
},

_replaceTextOnly: function(e) {
    for (var t, i = n.length, r = 0; i > r; r++) {
        for (t = e[r], i.innerHTML = TD.emoji.parse(t.nodeValue), i = document.createDocumentFragment(); i.firstChild) i.appendChild(t.firstChild);
    } t.parentNode.replaceChild(t, i)
},

_emojiify: function(e) {
    var t;
    return TD.emoji.test(e) & (D.innerHTML = e, t = [], this._parseTextOnly(D, t), this._replaceTextOnly(t, e) || (e = D.innerHTML, D.innerHTML = ""), e &
    return e
},

updateEntities: function(e, t) {
    var i, n, r, s, o, a, c, u = ttwtr.txt.modifyIndicesFromUnicodeToUTF16;
}

}, TD.emoji = function(e) {
    var t = {
        theme: "",
        path: "/web/assets/emoji",
        unified: {},
        parse: function(e) {
            return e.replace(/\u{FE0F}/g, "")
        },
        place: function(e, i, n) {
            var r = 1 === e.length & "" === n.charAt(i + 1) ? "" : "";
            return TD.ui.template.render("text/emoji", {
                alt: e + r,
                src: t.path + "" + t.theme + "" + t.unified[e] + ".png"
            })
        },
        test: function(e) {
            var i = t.re.test(e);
            return t.re.lastIndex = 0, i
        }
    }
// Replace emoji characters in the tweet with <img> tags
var emojified = this.emoji.parse(tweet.nodeValue);

// Make a new div, set innerHTML to the emojified value
newDiv.innerHTML = emojified;  // DANGER!!

// DOM surgery to replace the original tweet text with emojified
var i = document.createDocumentFragment();
while (newDiv.firstChild) {
    i.appendChild(newDiv.firstChild);
}
tweet.parentNode.replaceChild(i, tweet);  // script executes :(  

What happened?
wow great podcast. check out this one too!
wow great podcast. check out this one too!
Solution:
Tell the browser, “Only allow resources of Type X from Origin Y (and/or disallow inline scripts) on this page.”
**Content Security Policy**

1. Set your Content Security Policy as an HTTP Header.
   
   ```
   Content-Security-Policy:
   img-src *.cdn.example.com;
   script-src 'self' https://apis.google.com
   ```

2. Profit:

   ![Developer Tools - http://127.0.0.1:8000/csp.html](image)

   Refused to load the script 'http://evil.com/evil.js' because it violates the following Content Security Policy directive: "script-src 'self' https://apis.google.com".
I AM AWESOME

<table>
<thead>
<tr>
<th>document-uri</th>
<th>referrer</th>
<th>violated-directive</th>
<th>blocked-uri</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://74.6.34.39/test/CPLdpthNghLwzMxl">http://74.6.34.39/test/CPLdpthNghLwzMxl</a></td>
<td>script-src <a href="https://code.jquery.com">https://code.jquery.com</a> 'nonce-abc'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refused to execute inline script because it violates the following Content Security Policy directive: "script-src https://code.jquery.com 'nonce-abc'". Either the 'unsafe-inline' keyword, a hash ('sha256--aCbowRAQ1f8HO-Va4JssVlMgYbuKCdQlWYqKgCS-T8='), or a nonce ('nonce--...') is required to enable inline execution.
Setting up CSP

1. Run in report-only mode to determine the minimally-permissive policy for your site
2. Switch to enforce mode after “enough” testing
Referrers are annoying.
Super Secret Design Document

blah blah blah this is a super secret design doc.

To build the app, download this awesome tool.

August 2015 Release Plan
Web vulnerability affecting shared links

Aditya Agarwal  |  May 5, 2014  |  236 comments
Google Drive update to protect to shared links

Posted: Friday, June 27, 2014

Posted by Kevin Stadmeyer, Technical Program Manager

At Google, ensuring the security of our users is a top priority, and we are constantly assessing how we can make our services even more secure. We recently received a report via our Vulnerability Reward Program of a security issue affecting a small subset of file types in Google Drive and have since made an update to address it.

This issue is only relevant if all of the following apply:
- The file was uploaded to Google Drive
- The file was not converted to Docs, Sheets, or Slides (i.e. remained in its original format such as .pdf, .docx, etc.)
- The owner changed sharing settings so that the document was available to “Anyone with the link”
- The file contained hyperlinks to third-party HTTPS websites in its content
CSP to the rescue (again)

Set the “referrer” CSP directive to one of:
- “no-referrer”
- “no-referrer-when-downgrade” (default)
- “origin”
- “origin-when-cross-origin”
- “unsafe-url”
Why don’t people use CSP?

- Boring name
- ???
Solution

**BATSHIELD:**
Back-Acronymed
Trustworthy
Secure Helper
Internet-Enabled
Lightweight Defense

Sponsored by: Google Adwords

[Image of a hand-drawn design with a bat and shield, labeled BATSHIELD creators.
Without HTTPS, websites cannot guarantee:

- Server authentication
- Data confidentiality
- Data integrity
June, 2015
Problem #2:
Setting up SSL is usually tedious and costs $.
Purchasing a Signed Certificate from a Certificate Authority (CA)

You can also purchase a certificate directly from a Certificate Authority (CA) and install it in to your DreamHost panel. To do this, you'll need a Certificate Signing Request (CSR) which can be found in your panel.

The following steps explain how to obtain this CSR in your panel:

1. Review the Adding Secure Hosting (self-signed certificate) section above to add Secure Hosting and a self-signed certificate to your domain.
2. Go to the (Panel > Domains > Manage Domains) page.
   The 'Manage Domains' page opens:
3. To the right of your domain, click on the Certificates button.
4. When the 'Secure Hosting' page opens, click the 'Manual configuration' radio button to expose the current certificate information.

   There are several large text fields on this page:

   ![Certificate Settings for dhwiki.dreamhosters.com](image)

5. COPY (do NOT cut) the text from the Certificate Signing Request field box.
6. Paste the text into the order form from whichever Certificate Authority you'd like to purchase your signed SSL certificate.

--BEGIN CERTIFICATE REQUEST----
MIICcBICAدوCAAwggxzdJABgNVBAYTAVTimwEQYDVQQgXzKwZzZmYzIwMDoXDA8BQgNVBAgT
...
75M+Si8+N4IzAmUvP+YmCjk+CjyW0cn6aeY+41tl0wskb9a1mnnm/zX0b8pK0

---END CERTIFICATE REQUEST---

When purchasing a signed SSL certificate, you must specify the server type:
- To use the SSL certificate on DreamHost's servers, specify the server type as: Apache 2.0 w̄MOD_SSL
- Once you successfully complete your purchase, the CA will send you the signed SSL certificate; you can then replace the self-signed SSL certificate with this signed SSL certificate in your panel.
- For more information, see the instructions in the following Installing a Sign Certificate you've already purchased section.
Let’s Encrypt

Solution:
Start a certificate authority that gives out free certificates.
Let’s Encrypt

... And provide a server package that automates domain validation, certificate issuance, SSL configuration, and certificate renewal.

Technical details: https://github.com/letsencrypt/acme-spec/
Congratulations! You have successfully enabled https://foo.com/lets-ssl/.
Let’s Encrypt

https://letsencrypt.org

Launching the week of Sept. 14, 2015
On the Horizon:

- Subresource Integrity
- Fixing mixed content
- Privacy/security standards for W3C specs
- Restricting new web features to HTTPS only
- User-to-user encrypted messaging on the web
Thanks!
yan@eff.org
yyy@yahoo-inc.com
Twitter: @bcrypt