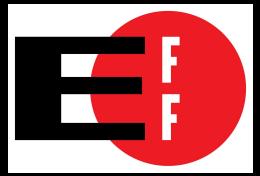
Weird Tricks to Improve Web Security 10000000%

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.

how 2 pwn websites

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.

Frederic Jacobs
@FredericJacobs

Follow

The Mac application for Tweetdeck does not appear to be vulnerable to the XSS. Confirmed in Chrome though. <script>alert("Yo!");</script> ❤

Reply Retweet Favorite ••• More

RETWEETS 36 FAVORITES 16

8:55 AM - 11 Jun 2014

The Great Tweetdeck XSS of June 2014

```diff
@@ -6301,18 +6299,8 @@ function(){
         transform: function(e, t) {
             return e ? (t = t || {}, this.updateEntities(e, t), e = this.linkify(e, t), this.emojify(e)) : ""
         },
-        _parseTextOnly: function(e, t) {
-            for (var i, n = e.childNodes, r = n.length, s = 0; r > s; s++) i = n[s], 3 === i.nodeType && TD.e
moji.test(i.nodeValue) ? t.push(i) : i.hasChildNodes() && this._parseTextOnly(i, t)
-        },
-        _replaceTextOnly: function(e) {
-            for (var t, i, n = e.length, r = 0; n > r; r++) {
-                for (t = e[r], w.innerHTML = TD.emoji.parse(t.nodeValue), i = document.createDocumentFragment
(); w.hasChildNodes();) i.appendChild(w.firstChild);
-                t.parentNode.replaceChild(i, t)
-            }
-        },
         emojify: function(e) {
-            var t;
-            return TD.emoji.test(e) && (D.innerHTML = e, t = [], this._parseTextOnly(D, t), this._replaceText
Only(t), e = D.innerHTML, D.innerHTML = ""), e
+            return e
         },
         updateEntities: function(e, t) {
             var i, n, r, s, o, a, c, u = twttr.txt.modifyIndicesFromUnicodeToUTF16;
@@ -6893,36 +6881,39 @@ function(e, t) {
         return n
     }, t
 }), TD.emoji = function(e) {
-    var t = {
-        theme: "",
-        path: "/web/assets/emoji",
-        unified: {},
-        parse: function(e) {
-            return e.replace(t.re, t.place).replace(/\uFE0F/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
-        }
```

359,34 60%

What happened?

```
// 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.hasChildNodes()) {
  i.appendChild(newDiv.firstChild);
}
tweet.parentNode.replaceChild(i, tweet);  // script executes :(
```

yan Post author

June 22, 2015 at 18:12

wow great podcast. check out this one too!

Edit

Reply ↓

You may use these HTML tags and attributes: `<a href="" title="">` `<abbr title="">` `<acronym title="">` `<b>` `<blockquote cite="">` `<cite>` `<code>` `<del datetime="">` `<em>` `<i>` `<q cite="">` `<s>` `<strike>` `<strong>`

Post Comment

https://zyan.scripts.mit.edu/blog/pseudorandom-podcast-series-episode-1/#comment-432

discrete blogarithm 8 1 New Edit Post

yan Post author

June 22, 2015 at 18:12

wow great podcast. check out this one too!

Edit

Reply |

The page at https://zyan.scripts.mit.edu says:

wordpress_test_cookie=WP+Cookie+check; wp-settings-1=editor%3Dtinymce%26imgsize%3Dfull%26libraryContent%3Dbrowse%26advImgDetails%3Dshow; wp-settings-time-1=1434996551

OK

avascript:alert(document.cookie)

Elements Network Sources Timeline Profiles Resources Audits Console HTTPS Everywhere

```
<!-- #comment-## -->
▼<li class="comment byuser comment-author-yan bypostauthor odd alt thread-odd thread-alt depth-1" id="li-comment-432">
  ▼<article id="comment-432" class="comment">
    ▶<header class="comment-meta comment-author vcard">…</header>
     <!-- .comment-meta -->
    ▼<section class="comment-content comment">
      ▼<p>
          "wow great podcast. check out "
          <a href="javascript:alert(document.cookie)" rel="nofollow">this one too</a>
          "!"
       </p>
      ▶<p class="edit-link">…</p>
     </section>
```

Styles Computed Event Listeners DO

```
element.style {
}
media="all"
.entry-content a:visited, .comment-content a:visited {
   color: #9f9f9f;
}
media="all"
a {
   outline: ▶none;
   color: #21759b;
}
```

Content Security Policy

Solution:

Tell the browser, “Only allow resources of Type X from Origin Y (and/or disallow inline scripts) on this page.”

Content Security Policy

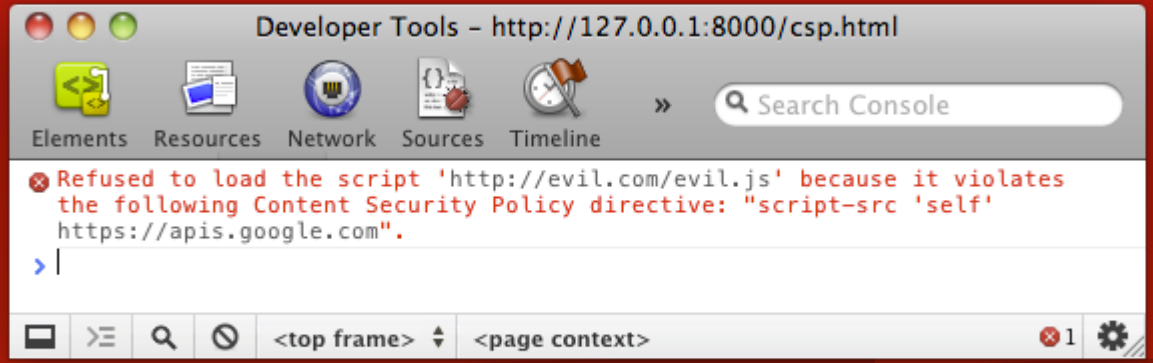
1. Set your Content Security Policy as an HTTP Header.

Content-Security-Policy:

*img-src *.cdn.example.com;*

script-src 'self' https://apis.google.com

2. Profit:



csptester.io

Content Security Policy Tester

A quick and easy way to test CSP behavior on modern browsers

CSP header

```
script-src https://code.jquery.com 'nonce-abc'
```

☐ Report only

HTML code

```
<div>Hello world</div>
<script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
<script>$('div').text('I SUCK')</script>
<script nonce='abc'>$('div').text('I AM AWESOME')</script>
```

csptester.io/test

Content Security Policy Tester

A quick and easy way to test CSP behavior on modern browsers

WebKit CSP Tests

CSP Quick Reference

Help

Home

I AM AWESOME

	document-uri	referrer	violated-directive	blocked-uri
1	http://74.6.34.39/test/CPLdpthNghLwzMxl		script-src https://code.jquery.com 'nonce-abc'	

Elements Network Sources Timeline Profiles Resources Audits Console » 1

<top frame> Preserve log

```
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-_aCbowRAQifb8HO-Va4JsVlMgybuKCdQlwYqKgCS-T8='), or a nonce ('nonce-...') is required to enable inline execution.
```

CPLdpthNghLwzMxl:3

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

Github's CSP header

```
Content-Security-Policy: default-src *; script-src assets-cdn.github.com collector-cdn.github.c
om; object-src assets-cdn.github.com; style-src 'self' 'unsafe-inline' 'unsafe-eval' assets-cd
n.github.com; img-src 'self' data: assets-cdn.github.com identicons.github.com www.google-analy
tics.com collector.githubapp.com *.githubusercontent.com *.gravatar.com *.wp.com; media-src 'no
ne'; frame-src 'self' render.githubusercontent.com gist.github.com www.youtube.com player.vime
o.com checkout.paypal.com; font-src assets-cdn.github.com; connect-src 'self' live.github.com w
ss://live.github.com uploads.github.com status.github.com api.github.com www.google-analytics.c
om github-cloud.s3.amazonaws.com
```

CSP + Refer(r)ers

Referrers are annoying.

. [US] https://github.com/diracdeltas/supersecret/blob/feature/fb-integration/mail_app/react-docs.md#august-2015-release-plan

This repository Search Pull requests Issues Gist

diracdeltas / supersecret

Unwatch 1

branch: feature/fb-int... supersecret / mail_app / react-docs.md

diracdeltas 12 seconds ago Add more details

1 contributor

9 lines (5 sloc) 0.194 kB

Raw Blame History

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

OK!!
Let's click on this!

discrete blogarithm | Yan's blog.

https://zyan.scripts.mit.edu/blog/

discrete blogarithm

Yan's blog.

HOME ABOUT

Inspector | Console | Debugger | Style Editor | Performance

Net | CSS | JS | Security | Logging | Clear

```
document.referrer
"https://github.com/diracdeltas/supersecret/blob/master/mail_app/react-docs.md"
```

Dropbox, May 2014

Dropbox Blog

Topics ▾ Subscribe ▾ Dropbox blogs ▾

Web vulnerability affecting shared links

Aditya Agarwal | May 5, 2014 | 236 comments

563 8 144 84

Google, June 2014

Google | Online Security Blog

The latest news and insights from Google on security and safety on the Internet

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”

Source: http://caniuse.com/

IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android
5.5	2	4	3.1	9	3.2	5.0-8.0	2.1	7	10	42	38	10	9.9
6	3	5	3.2	9.5-9.6	4.0-4.1		2.2	10	11			11	
7	3.5	6	4	10.0-10.1	4.2-4.3		2.3		11.1				
8	3.6	7	5	10.5	5.0-5.1		3		11.5				
9	4	8	5.1	10.6	6.0-6.1		4		12				
10	5	9	6	11	7.0-7.1		4.1		12.1				
11	6	10	6.1	11.1	8		4.2-4.3						
Edge	7	11	7	11.5	8.1-8.3		4.4						
	8	12	7.1	11.6	9		4.4.3-4.4.4						
	9	13	8	12			40						
	10	14	9	12.1									
	11	15		15									
	12	16		16									
	13	17		17									
	14	18		18									
	15	19		19									
	16	20		20									
	17	21		21									
	18	22		22									
	19	23		23									
	20	24		24									
	21	25		25									
	22	26		26									
	23	27		27									
	24	28		28									
	25	29		29									
	26	30		30									
	27	31		31									
	28	32		32									
	29	33											
	30	34											
	31	35											
	32	36											
	33	37											
	34	38											
	35	39											
	36	40											
	37	41											
	38	42											
	39	43											
	40	44											
	41	45											
		46											

Why don’t people use CSP?

- Boring name
- ???

Solution

BATSHIELD:

Back-**A**cronymed
Trustworthy
Secure **H**elper
Internet-**E**nabled
Lightweight **D**efense

HTTPS

Without HTTPS, websites cannot guarantee:

- Server authentication
- Data confidentiality
- Data integrity

June, 2014

www.quora.com/

Allow www.quora.com to run "Adobe Flash"? Continue Blocking Allow...

Quora

Sign up to read Quora

yan ••••••••••••••••••• Login

No account found for this email. Retry, or Sign up for Quora.

By logging in you indicate that you have read and agree to the Terms of Service.

Sign Up With Google

Console HTML CSS Script DOM Net Cookies Page Speed Search within Net panel

Clear Persist All HTML CSS JavaScript XHR Images Plugins Media Fonts

POST server_call_	200 OK	quora.com	129 B	192.33.31.50:80	352ms
POST server_call_	200 OK	quora.com	129 B	192.33.31.50:80	1.65s
POST server_call_	200 OK	quora.com	136 B	192.33.31.50:80	649ms

Params Headers Post Response JSON Cache Cookies

Parameters application/x-www-form-urlencoded Do not sort

__e2e_action_id	du8hkifibq
__vcon_json	["hmac","nZJIEhRrZfyXGv"]
__vcon_method	do_login
_lm_transaction_id	0.6176264159256558
_lm_window_id	dep186-1166192661471933980
formkey	17faa36c0f3bf2fa0c778572d87842c0
js_init	{}
json	{"args":[],"kwargs":{"email":"yan","password":"somestringpassword","passwordless":1}}
postkey	9ca83f552816f040b6bba34050f0a888
referring_action	index
referring_controller	index
window_id	dep186-1166192661471933980

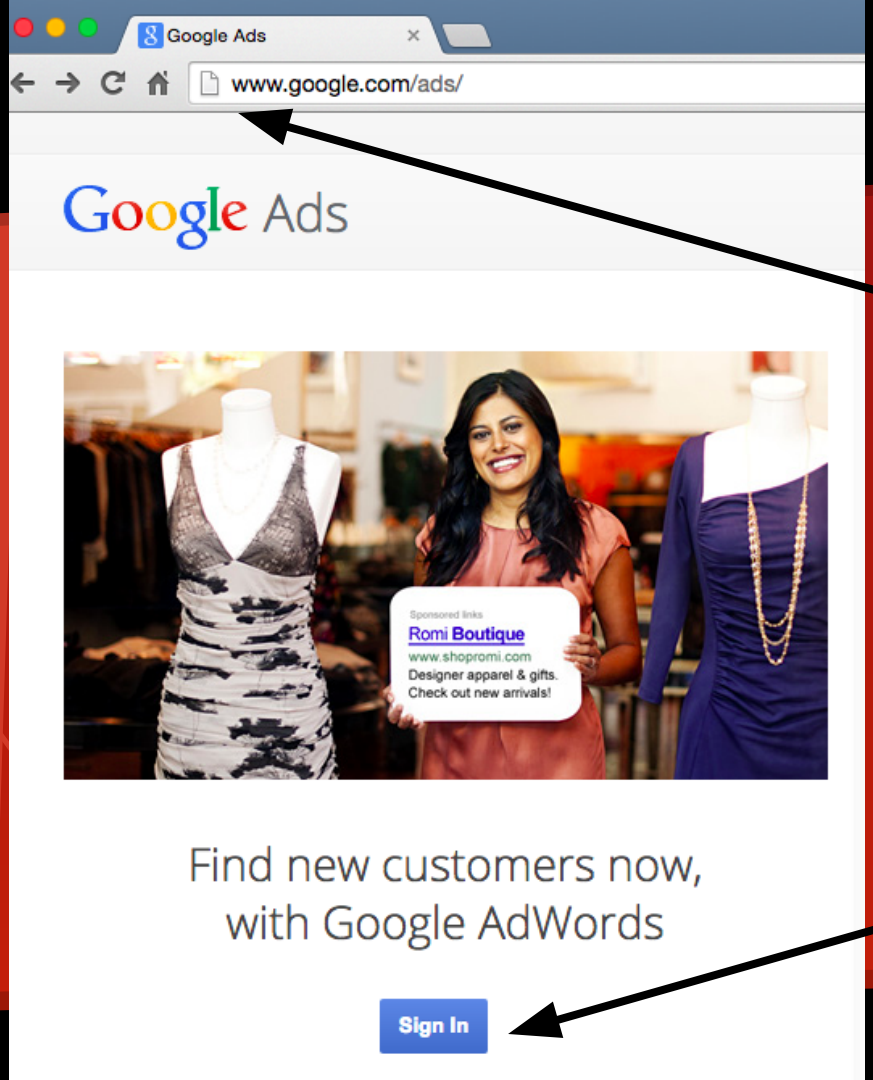
Source

json=%7B%22args%22%3A%5B%5D%2C%22kwargs%22%3A%7B%22email%22%3A%22yan%22%2C%22password%22%3A%22somestringpassword

POST http://www.quora.com/webnode2/server_call_POST?__instart__ 200 OK 639ms -d476b1...018a.js (line 9)

POST http://www.quora.com/webnode2/server_call_POST?__instart__ 200 OK 352ms -d476b1...018a.js (line 9)

June, 2015


Google Ads
www.google.com/ads/
Google Ads
Sponsored links
Romi Boutique
www.shopromi.com
Designer apparel & gifts.
Check out new arrivals!
Find new customers now,
with Google AdWords
Sign In

(ﾉ°□°)ﾉ︵ ┻━┻

:(

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.

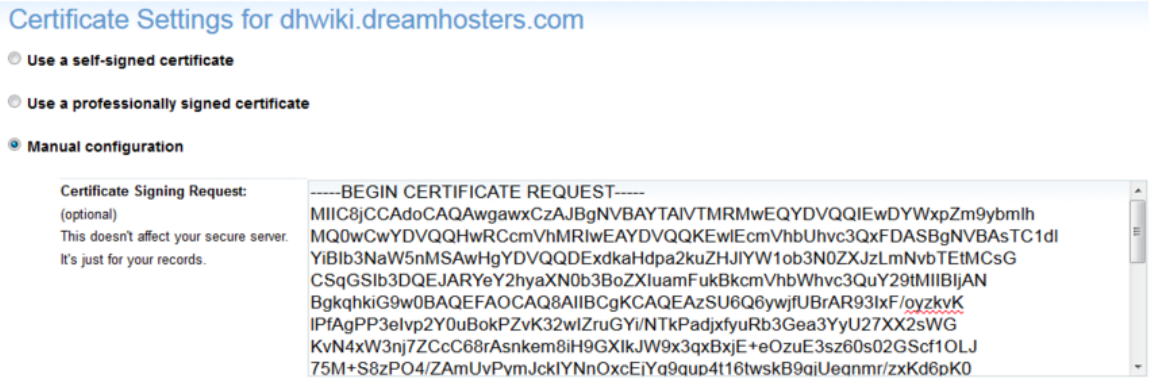
The following steps explain how to obtain this CSR in your panel:

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

Use a self-signed certificate

Use a professionally signed certificate

Manual configuration

Certificate Signing Request:
(optional)
This doesn't affect your secure server.
It's just for your records.

-----BEGIN CERTIFICATE REQUEST-----
MIIC8jCCAdoCAQAwgawxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwDYWxpZm9ybmlh
MQ0wCwYDVQQHwRCcmVhMRIwEAYDVQQKEwlEcmVhbUhvc3QxFDASBgNVBAsTC1dl
YiBlb3NaW5nMSAwHgYDVQQDExdkaHdpa2kuZHJlYW1ob3N0ZXJzLmNvbTEtMCsG
CSqGSIb3DQEJARYeY2hyaXN0b3BoZXIuamFukBkcmVhbWhvc3QuY29tMIIBIjAN
BgkqhkiG9w0BAQEFAOCAQ8AIIBCgKCAQEAzSU6Q6ywjfUBrAR93IxF/oyzkvK
lPfAgPP3eIvp2Y0uBokPZvK32wlZruGYi/NTkPadjxfyuRb3Gea3YyU27XX2sWG
KvN4xW3nj7ZCcC68rAsnkem8iH9GXIkJW9x3qxBxjE+eOzuE3sz60s02GScf1OLJ
75M+S8zPO4/ZAmUvPymJcklYNnOxcEiYq9qup4t16twskB9qiUegnmr/zxKd6pK0

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.

Important:

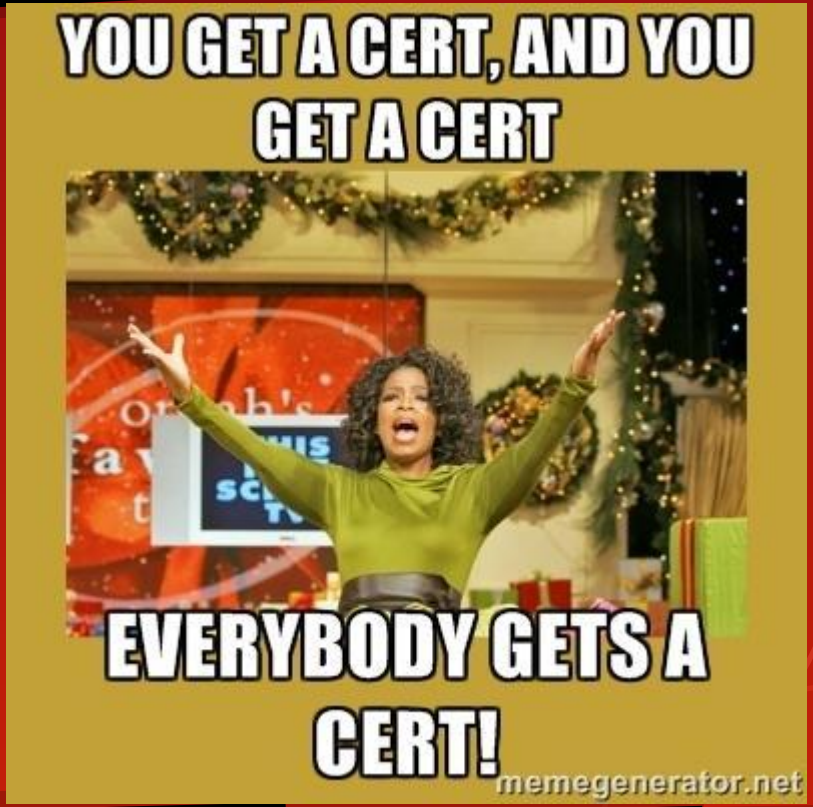
When purchasing a signed SSL certificate, you must specify the server type:

- To use the SSL certificate on DreamHost's servers, specify the server type as: `Apache 2.X w/MOD_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:

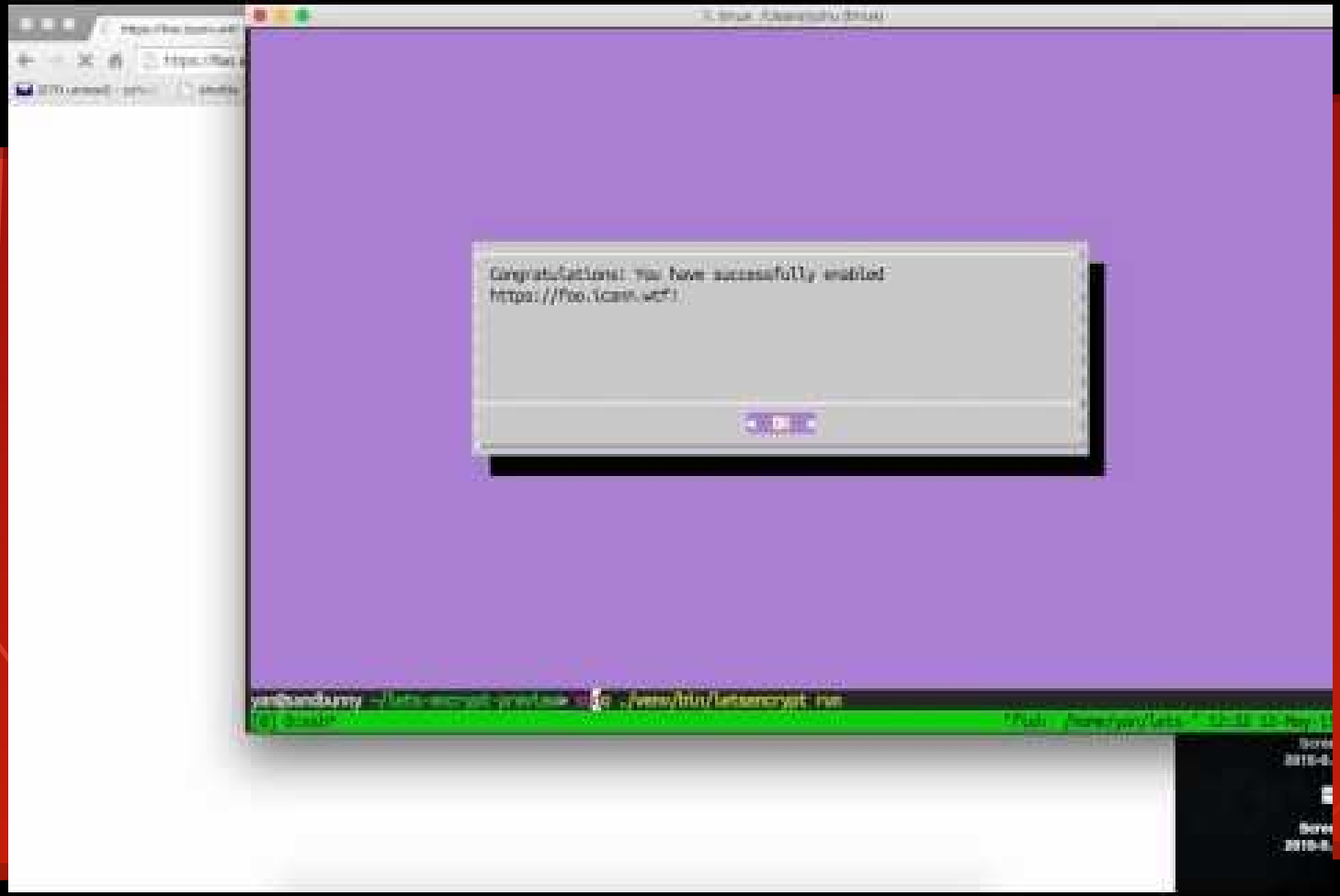
Start a certificate authority that gives out free certificates.



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.icann.wtf!

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