Let’s be a dork and read .js files

Helping those new to hunting: What is dorking and learning what you’re looking for with .js files

What exactly is dorking…?

- Using public search engines such as Google, GitHub, Yahoo, Bing, Shodan, to find public data about your target. They do the spidering for you, just ask them the right questions!

site:example.com inurl:& - Finding parameters, scrape & try these on every endpoint you discover. (Remember past streams, devs re-use code. Old code etc)

Site:example.com ext:php (jsp, asp, aspx, xml, txt) - Discovering content on their sites, maybe sometimes old files that were indexed a long time ago. Helpful if to get an insight into what type of payloads/bugs you should focus on. (For example: See lots of php? Probably going to find XSS quite easily)

Site:example.com inurl:admin (login, register, signup, unsubscribe, redirect, returnUrl, Get creative!!! The possibilities are endless. Ask and you shall receive) - Finding functionality to play with.

On Shodan, simply search for their IP range as well as Ssl:".example.com", Org:"Example"
More: https://help.shodan.io/the-basics/search-query-fundamentals

Your search - **site:mail.ru inurl:&** - did not match any documents.

Suggestions:

- Make sure that all words are spelled correctly.
- Try different keywords.
- Try more general keywords.
- Try fewer keywords.

In order to show you the most relevant results, we have omitted some entries very similar to the 90 already displayed.
If you like, you can repeat the search with the omitted results included.

What about GitHub?

- Searching through devs code and finding potential exposed staging servers, internal api's, (think swagger). Even test code sometimes. Again it's as simple as asking the right questions.

"Company" stage (staging,stg,dev,prod,qa, swagger) - Have any devs pushed code containing staging information or any swaggerui exposed? Even if these domains can't be accessed from outside their network, save it for SSRF in future?

"Company" apiKey (apiSecret,x-api-key, apidocs, /api/, /internal/api) - Any exposed api keys? Find API docs and see what the key is used for. Maybe an internal key is leaked which bypasses rate limiting for example. (Had a case in which login flow used ?client_id= which was always same. Found another on github which bypassed 2FA)

"Company" [keyword] - Look for specific features/services such as login, signup, registration, admin, administrator, appspot (staging servers?), firebase, password, password, testuser, testing)

Get creative!!! :)

.JS files: What exactly are in them?

- .js files contain code which help a website function. For example when clicking on a link, they may have onclick=”runCode()” which executes javascript based on your click. This can be used for tracking purposes, setting certain cookies, checking parameters. .js files play a big part in how a website functions.

- The process when looking for .js files is as simple as:

  Visit your target, right click -> view source. CTRL + F and search for “.js”
  Browse each .js files and search for certain keywords: api, internal, url:, var =, //, https://, company.com, location.search, parameter

  (This is manual. You can automate this with Burp, but that’s for another time. Lots of info out there already, this for people new:D)

- What you’re looking for:

  New parameters/variables. References to more /api/ calls. For example can you make these api calls without having the correct permissions? Think about functionality that lets you upgrade. Go into the .js files & find this code and try the API calls. Look for dev comments, new endpoints, new subdomains. Api keys.

Code packed? Use https://beautifier.io/ to beautify code so it’s readable

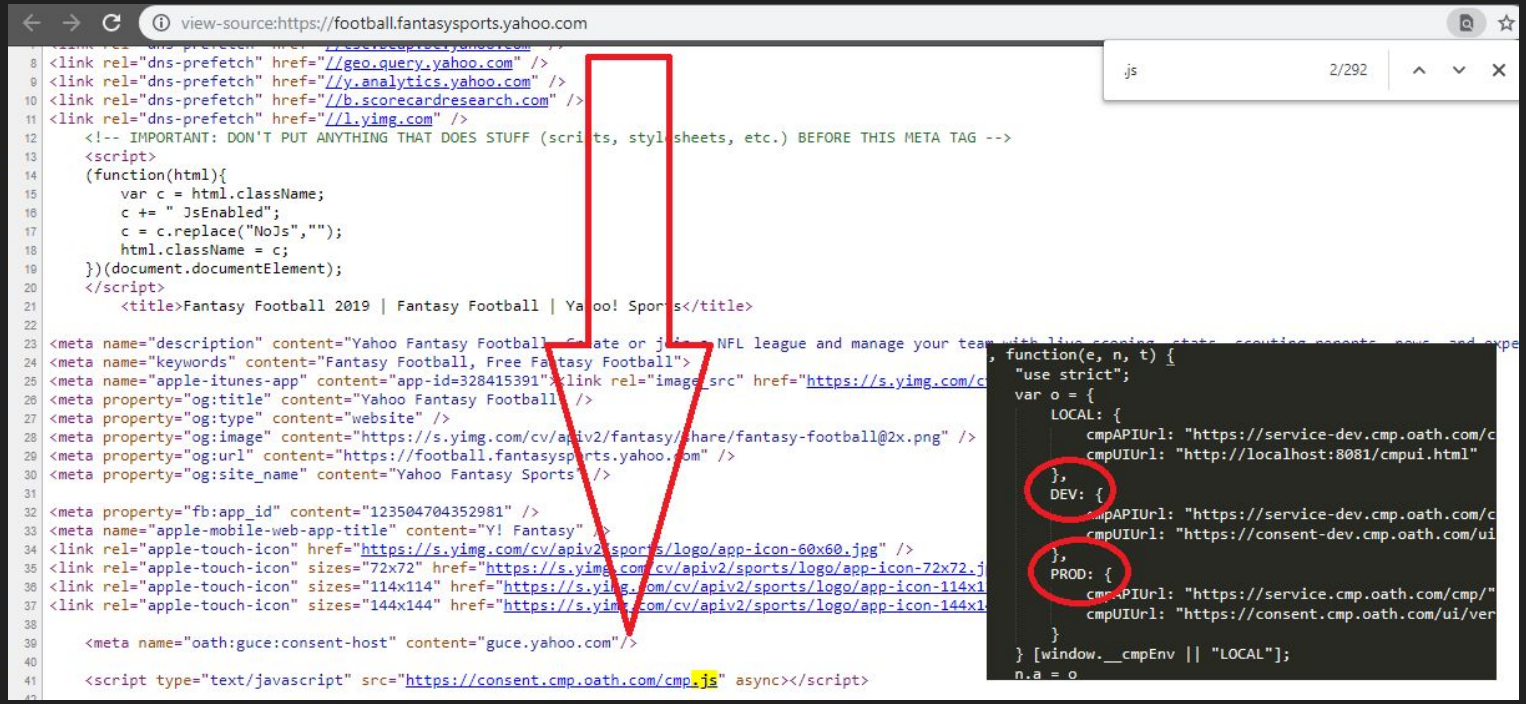
Some examples

- I could upgrade my account to three tiers, level 1, 2 and 3. Each one gave a different perk. One .js file contained *ALL* api requests for all tiers. Level 1 could access 2 & 3 without upgrading.

  Remember: .js files contain the code needed for a website to function. Even though you need to pay more to upgrade to access more features, this check will typically be done server-side. JS code is client-side and will be in the .js files regardless of your account status.

- I found references to internal staging subdomains if a certain cookie was set. Setting the cookie and visiting this endpoint revealed new /API/ calls being made which lead to an info leak.

- Google dorking found zero results for inurl:&. From viewing .js files I discovered “r_url” which acted as a redirect parameter on the login flow. The result? Token leak. Turns out the code was old and should not of been used anymore.




```
function LoyaltyMemberTrackingEvent() {
    if (void 0 !== dataLayer && Cookies.get("LoyaltyMember")) {
        var e = Cookies.get("LoyaltyMember");
        dataLayer.push({
            LoyaltyMember: e,
            event: "dataLayerUpdate"
        })
    }
}
```

```
return false;}},register:{name:'register',height:500,width:500,callback:function(params)
{    register.init({context:        .popup,redirect_url:params.redirect,async:true});}},movecopy:
```

Any questions? :)