【安全研究】恶意chrome扩展分析-Part1

原创：skytina 青藤实验室

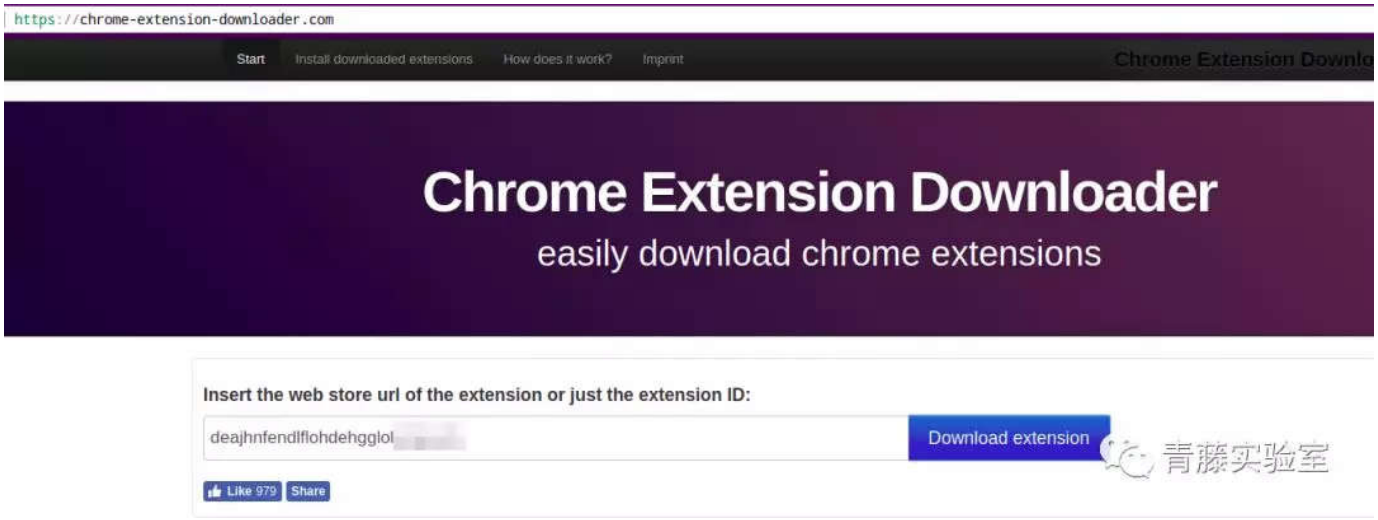
> Chrome浏览器支持通过扩展来丰富其功能，在Chrome的网上应用店中，除了有正经的扩展，也存在着一些“老不正经的”扩展。通过本篇文章，我们可以了解到如何开始分析Chrome浏览器扩展。

1. 获取恶意扩展

操作步骤如下:

1)获取Chrome扩展的ID。

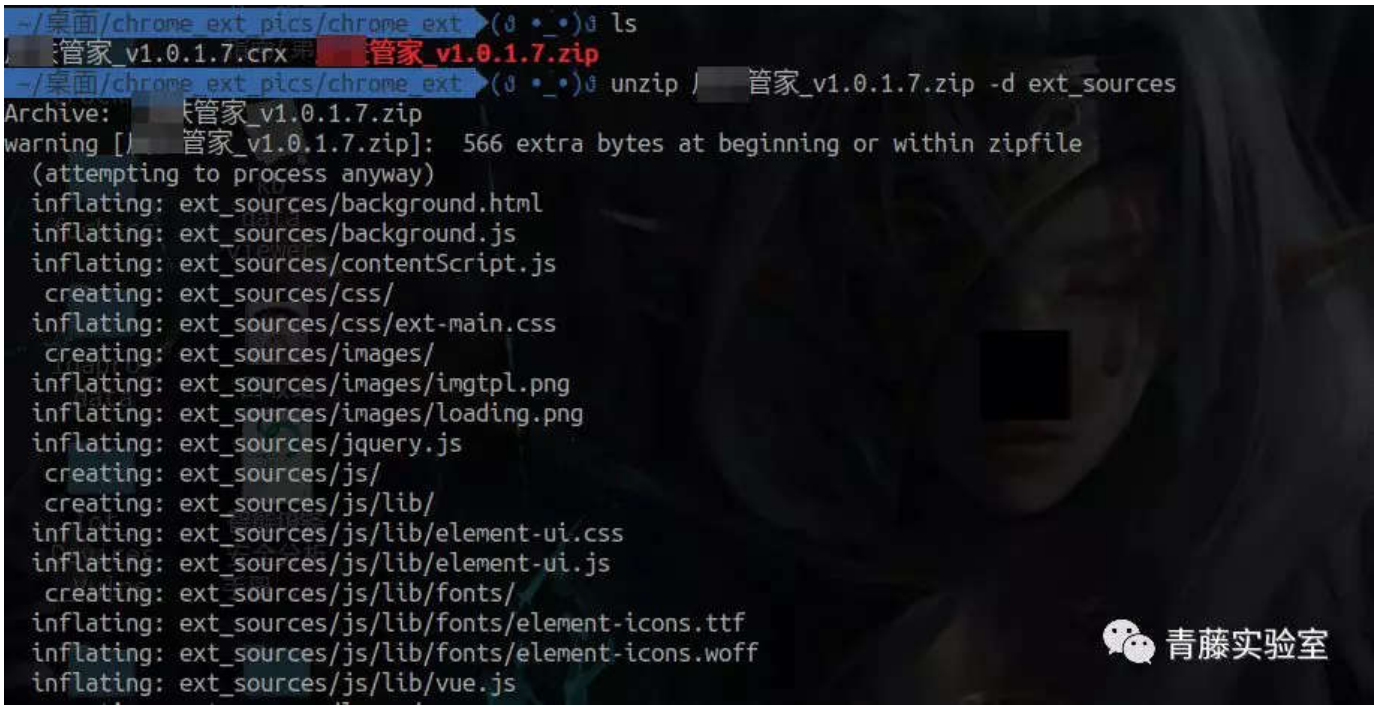
2)打开chrome-extension-downloader网站，输入扩展ID，点 **Download extension。**



2. 扩展解包

下载后的Chrome扩展后缀名为.crx，但其本质为一个zip压缩包，我们可以通过修

改其后缀名并使用相应的解压缩软件进行解包。



操作命令如下:

```
cp evil_ext.crx evil_ext.zip
unzip evil_ext.zip -d ext_sources
```

3.扩展分析入口

mainfest.json文件罗列了一个Chrome扩展的基本信息，我们通过这个了解到扩
的名称、版本信息，同时还能知道运行该扩展所需要的权限，不同功能对应的js文
件或者页面。
对于一个安全人员，我们在分析扩展的时候，需要重点关注mainfest.json文件的
以下字段信息。

- browser_action
  - default_popup：定义了点击扩展图标后的默认弹出内容，形式为HTML
    页面(如: popup.html)
- content_scripts
  - matches：定义了插入js文件的时机，默认为 "document_idle"
  - js：定义了要插入匹配页面的 JavaScript 文件列表，它们将按照数组中指
    定的顺序插入(如: ["jquery.js","contentScript.js"])
  - run_at：定义了哪些页面需要插入指定的内容脚本,支持正则表达式(如:
    http://*.baidu.com)

- permissions
  - 定义了扩展所需要的使用权限，不同的权限意味着你能够使用的不同的chrome.* API
- content_security_policy
  - 定义了内容安全策略(CSP)(如: script-src 'self'; object-src 'self')。
  - 注意: 对content_scripts内的脚本不起作用。
- background
  - persistent：定义扩展的后台页面，后台页面将由扩展程序系统生成，包含 scripts 属性中列出的每一个文件。
  - page：定义扩展的后台页面，内容为HTML页面(如: background.html)
  - scripts：当值为**false**的时候，background定义的page或者scripts为事件页面，只在需要的时候加载,不会在后台一直运行。

4. 从一个恶意扩展mainfest.json开始

通过步骤3,我们可以了解到一个浏览器扩展的分析应该先从其种的mainfest.json手，那么下面我们具体来看一个恶意扩展的mainfest.json文件，了解它的一些基本信息。

```
{
  "update_url": "https://clients2.google.com/service/update2/crx",
  "name": "x管家",
  "version": "1.0.1.7",
  "manifest_version": 2,
  "description": "x管家是一款可以改变网站主题样式的软件，不但可以改变样式，还可以在客户端内编辑样式代码，DIY自己的网页样式,让网站做到与众不同,只要浏览器安装了皮肤管家就可以使目标网站的样式进行更改",
  "icons": {
    ......
  },
  "browser_action": {
    ......,
    "default_popup": "popup.html"
  },
  "content_scripts": [
    {
      "js": [
        "jquery.js",
        "contentScript.js"
      ],
      "matches": [
        "http://*/*",
```

```
        "https://*/*"
      ],
      "run_at": "document_idle"
    }
  ],
  "permissions": [
    "http://*/*",
    "https://*/*",
    "storage",
    "unlimitedStorage",
    "webRequest",
    "webRequestBlocking",
    "tabs",
    "management",
    "nativeMessaging"
  ],
  "content_security_policy": "script-src 'self'  'unsafe-eval'; object-src 'sel
,
  "background": {
    "page": "background.html"
  },
  "web_accessible_resources": [
    "jquery.js"
  ]
}
```

4.1 browser_action

- default_popup
  - popup.html：点击右上角的扩展图标后默认的弹出页面，注意HTML不
    持内嵌的JavaScript代码执行。

4.2 content_scripts

- matches
  - http://*/*：匹配使用 https 协议的任意 URL。
  - https://*/*：匹配使用 http 协议的任意 URL。
- js
  - ["jquery.js","contentScript.js"]：结合matches，可知扩展往任意使用
    http以及https协议的页面依次注入"jquery.js","contentScript.js"。
- run_at
  - document_idle：在 "document_end" 和刚发生 window.onload 事
    这两个时刻之间插入js脚本,默认值。

4.3 permissions

- http://*/*与https://*/*

授予扩展访问所有使用http、https协议的主机权限。

- storage
  使您的应用能够访问 chrome.storage API(扩展可以通过chrome.storage储一些数据)。
- unlimitedStorage
  提供无限的存储空间，如果没有这一权限，扩展程序或应用的本地存储将限制在 5 MB 以内。
- webRequest
  监控与分析流量，可以实时地拦截、阻止或修改请求(该权限常用于实现代理功能、请求拦截功能的扩展，如Proxy SwitchyOmega、Postman)。但对于个修改网站样式的扩展来说，这样的权限显得**不正常**。
- webRequestBlocking
  允许你以阻塞方式使用网络请求 API(与webRequest权限配套出现)。
- tabs
  允许你获取Tab 的 url、title 和 favIconUrl 属性。
- management
  管理已经安装并且正在运行的扩展程序或应用。
- nativeMessaging
  使您的应用能够访问原生消息通信 API(内容脚本主要该API与扩展的后台或popup进行数据传递使用)。

4.4 content_security_policy

- script-src 'self' 'unsafe-eval';
  允许非内容脚本使用eval函数。

4.5 background

- page
  - background.html：扩展的背景页面为background.html。

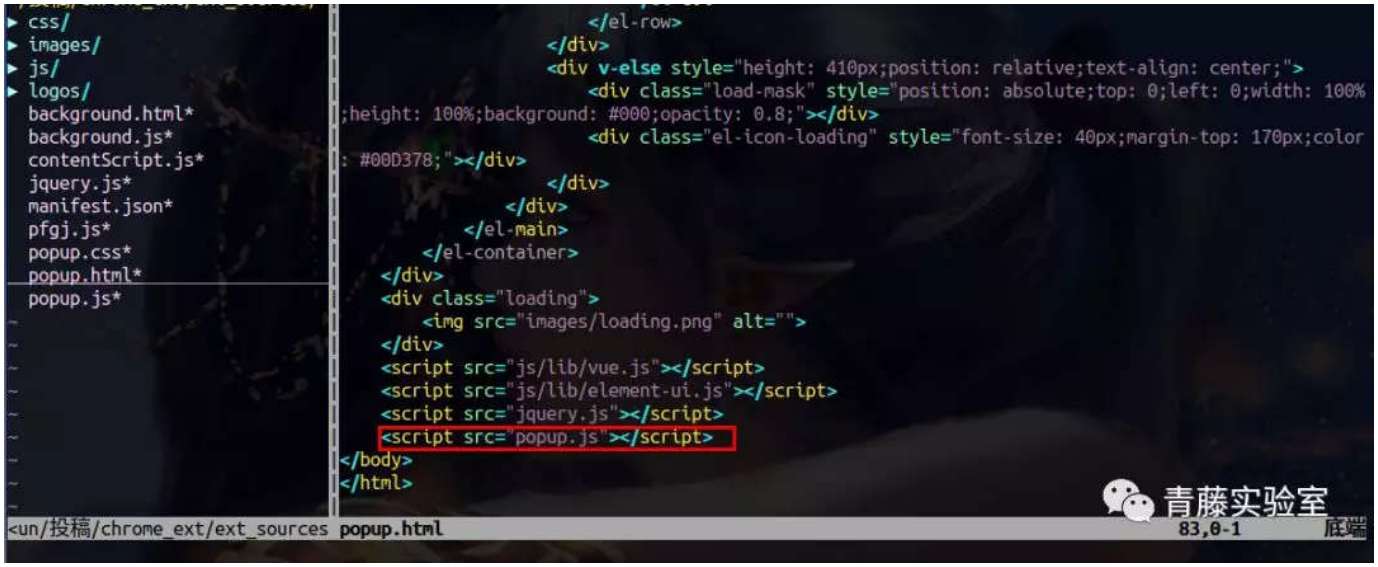
5 . 继mainfest.json之后的下一步

在我们分析完一个恶意扩展的mainfest.json文件之后，我们又该怎样去以一个合理的逻辑去分析扩展之间不同部分的关系，比如background对应的页面或者js脚本在什么时候执行，default_popup与content_scripts触发条件。

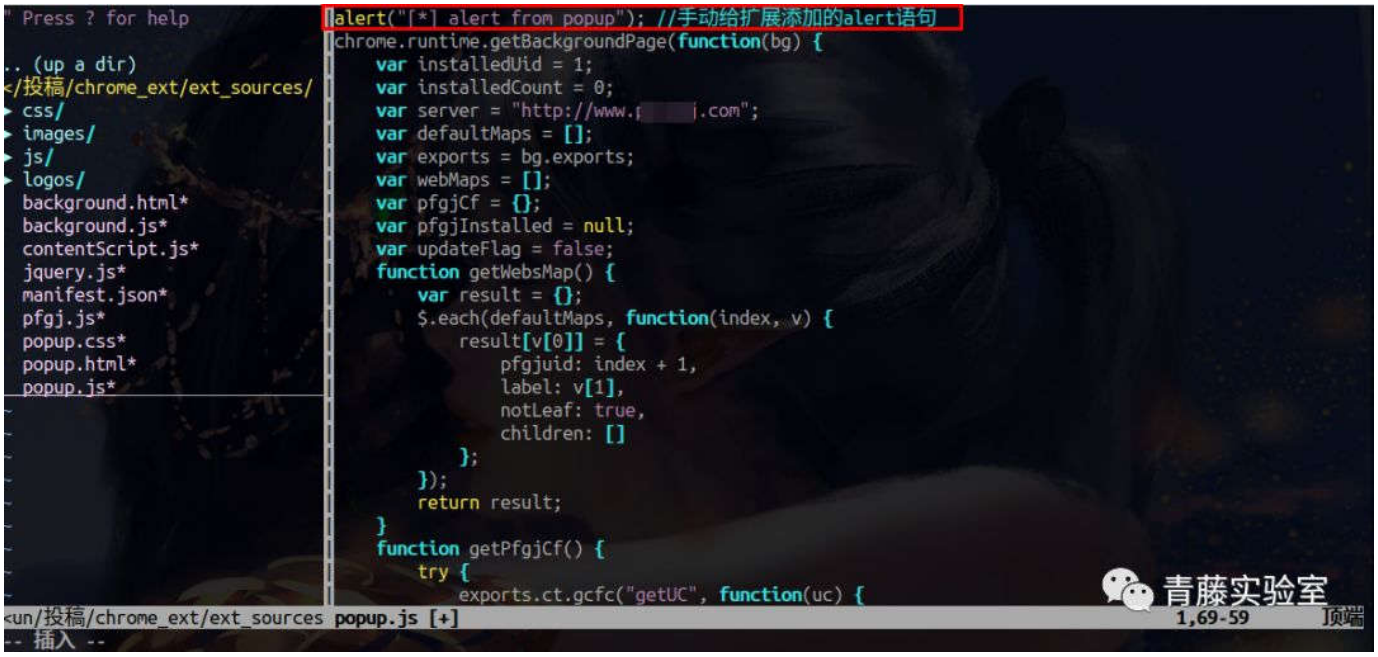
这里我们直接修改解包后的扩展源码，分别在background、content_scripts、default_popup的页面或者脚本中，加入弹窗函数，通过观**弹窗的顺序**，来了解扩展的不同脚本的加载顺序和触发时机。

5.1 修改default_popup

扩展的default_popup为popup.html，我们使用vim查看。

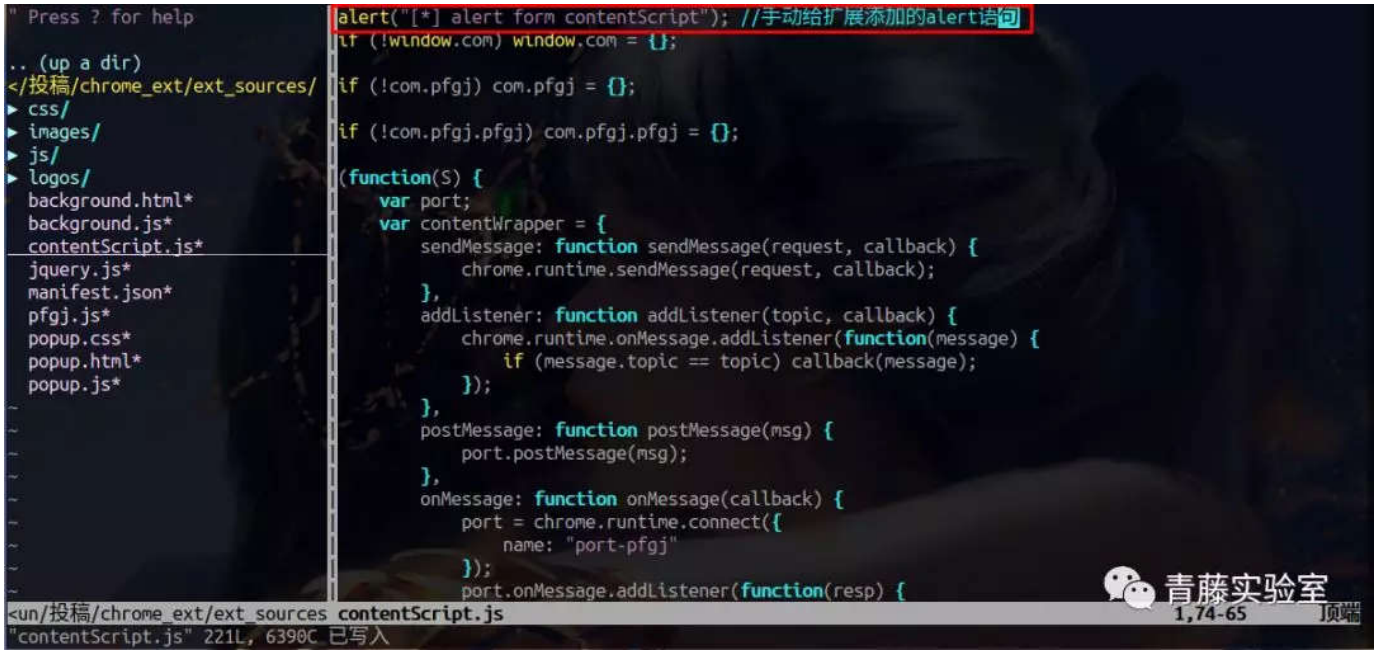


popup.html加载popup.js，我们在popup.js加入alert语句，当然你也可以在其js加入alert语句，只要popup.html会将它加载进去。



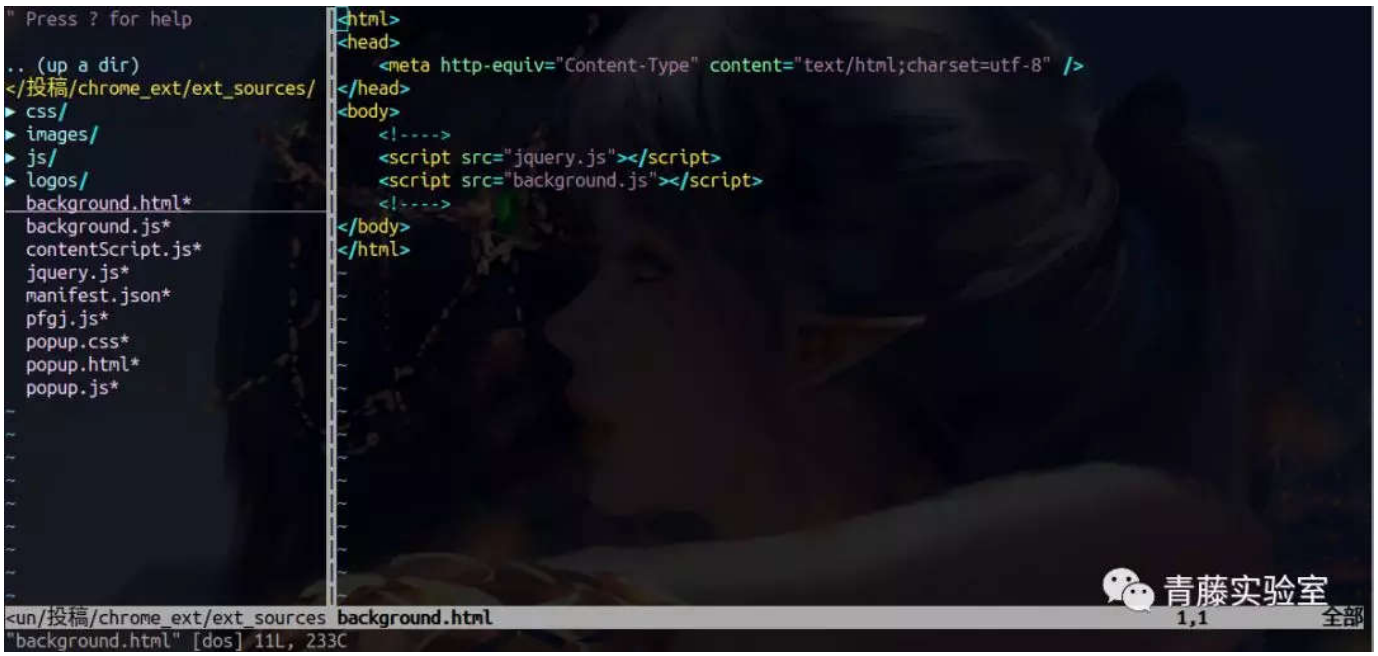
5.2 修改content_scripts

步骤四中我们了解到，扩展的的内容脚本。为"jquery.js"和"contentScript.js"，下面我们修改contentScript.js，加入alert语句。

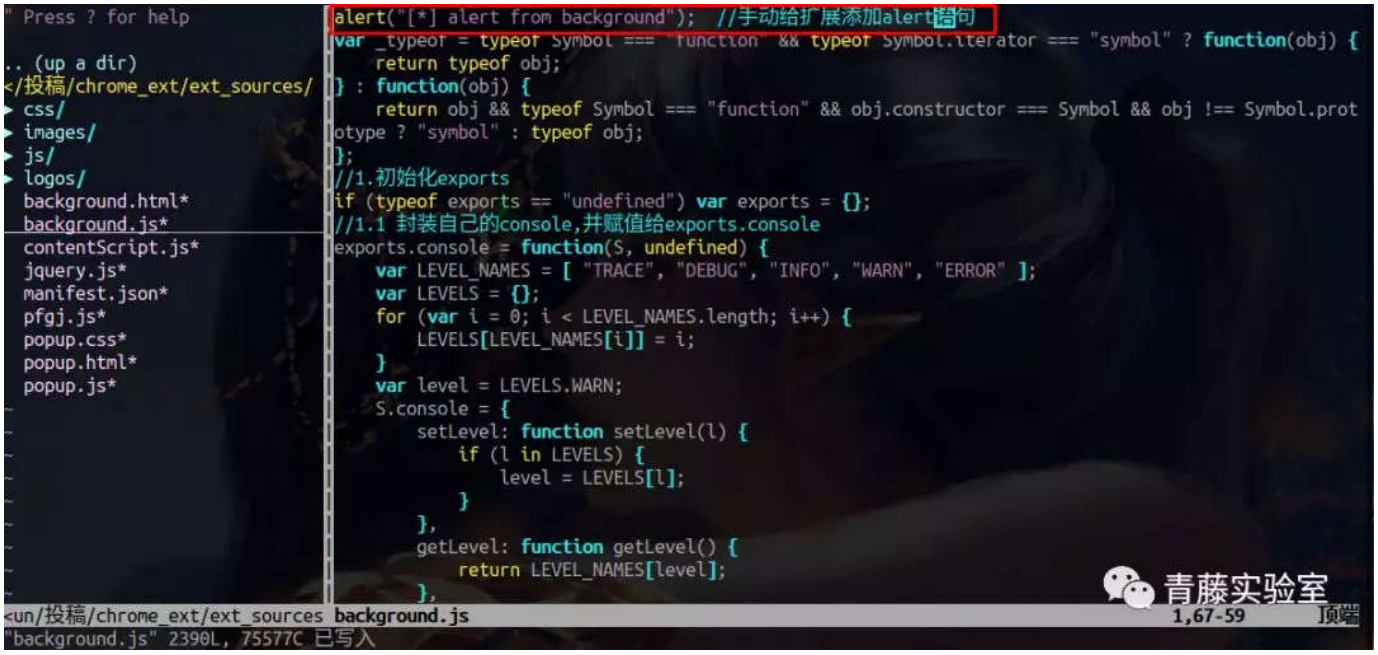


5.3 修改background

扩展的background为background.html,我们使用vim查看。



background.html加载background.js文件，我们在其中加入alert语句。



5.4 通过chrome加载已解压的扩展程序

- 浏览器地址栏中访问chrome://extensions
  - 将右上角的开发者模式打开
- 选择加载已解压的扩展程序，选择我们修改后的扩展源码目录

5.5 实际加载顺序

1)当我们**加载已解压的扩展程序**完成后，background里的alert语句直接被触发。

2)当我们访问任意使用HTTP协议或者HTTPS协议的网站时，content_scripts里的alert语句被触发。

3)当我们点击扩展图标时，default_popup里的alert语句被触发。

5.6 总结

通过5.5的分析，我们可以得出以下几点：

- 当扩展被加载时，就会触发background里面的内容(默认情况下加载是一次性的，除非我们手动刷新后台页面)。
- 被分析的恶意扩展会在chrome浏览器打开的所有使用HTTP或者HTTPS网站中注入内容脚本(即"jquery.js"和"contentScript.js")。
- 如果我们不去点击扩展图标，是不会触发default_popup的内容。

6. 后续工作

下一篇文章，我们将根据实际加载顺序去分析恶意扩展的真实行为，其中将会涉及到javascript、chrome调试、chrome开发者工具等内容。

参考链接

manifest.json文件作
用：https://developer.chrome.com/extensions/manifest