Ox01 题目的一些信息
http://58.229.183.26/html/Main.html?page=challenges
比赛的平台在资格赛结束之后还有开放，可以尝试去做做看。
web proxy 这个题目是 这样子的
□ description
==========

http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/

http://58.229.183.25/188f6594f694a3ca082f7530b5efc58dedf81b8d/

==========

给了两个 url 其实是用来减轻负载的 网站的内容都是一样的。
访问一下 发现是个可以提交 url 的页面

查看一下源码 发现里面又一行注释 给出了提示<!– admin/index.php –>

尝试访问
http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/admin/ 发现拒绝访问 403
所以猜测这个题目的目的就是要用那个 web_proxy 来代理 访问到 admin 内容 ./admin/index.php 里就可能会有 key 出来

Ox02 字符串过滤
那么我们就去利用那个代理去构造访问到 admin 的 url
http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=58.229.183.24%2F188f6594f694a3ca082f7530b5efc58dedf81b8d%2Fadmin%2F

发现访问不了 Access Denied

这里是用的外网的 ip 去访问的 可以尝试内网的 ip 或者是用回送地址 127.0.0.1 去访问
而且会对一些特殊的字符进行过滤 那么我们的想法就是要去绕过
被过滤的字符有这些 127.0.0.1 php http:// 当然 这些有些是可以用 url 编码来绕过的 比如 php 的 可以用 url:%70%68%70 来表示
当然 路径的构造 用/admin/就可以了 会直接访问到 index.php 上面的 所以这个 php 的过滤影响比较小

Ox03 回送地址绕过

影响比较大的是对 127.0.0.1 的过滤 这样在不知道他的内网的 ip 的情况下 只能尝试本机回送地址 可以尝试用 localhost 来绕过他

尝试过用 1 用%31 的 url 编码来绕过 即 127.0.0.%31 来表示 发现不是不解析访问不到 就是成 127.0.0.1 被过滤了 以为是浏览器的问题 用 FF 用谷歌都是一样的

在蓝莲花的 writeup 中还学到了一个技巧 127.0.1.1 这个 ip 也是可以用来表示本机的 ip 的

相关的一些资料可以搜索 127.0.0.0/8 ipv4 的 RFC：http://tools.ietf.org/html/rfc5735

简单来说就是 127.0.0.1——127.255.255.254 都是可以表示本机 ip 的 那么我们可以构造 127.0.1.1 或者是 127.0.0.2 这样子来绕过对于 127.0.0.1 的过滤

看到这里，我就想到了以前看到过的一个小技巧，ip 的缩写 127.0.0.1 可以缩写成 127.1 或者是 127.0.1 这样子 在 ip 不够 4 位时 会自动扩展 在中间增加 0 所以就可以用这个链接访问到 admin 的内容

http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=127.1%2F188f6594f694a3ca082f7530b5efc58dedf81b8d%2Fadmin%2F

看一下内容 发现又有新问题了

发现了可以看到页面响应头 但是 看不到页面的所有内容 只有两行

Ox04 Http Range

然后看到 writeup 又涨姿势了

%0A 的效果是换行符 然后通过这个换行符就可以想到来构造请求头中的各种信息，确实有灵光一闪的感觉 但其实就是经验少了，才没法想到的

后来查阅了相关资料之后才知道了 CRLF 注入攻击

HTTP Header CRLF Injection

许多网络协议，包括 HTTP 也使用 CRLF 来表示每一行的结束。这就意味着用户可以通过 CRLF 注入自定义 HTTP header，导致用户可以不经过应用层直接与 Server 对话。

HTTP header 的定义就是基于这样的”Key: Value”的结构，用 CRLF 命令表示一行的结尾。

“Location:”头用来表示重定向的 URL 地址，”Set-Cookie:”头用来设置 cookies。

如果用户的输入经过验证，其中存在 CRLF 的字符就可以被用来达到欺骗的目的。

http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=127.1/

http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=127.1/%0A%0A

返回的效果是不一样的 说明换行符对于这个请求是可以进行控制的

能够换行的话 容易想到 可以对 http 头进行一些控制 达到我们需要的效果
原来可以在 http 的响应头里增加一些信息来 控制查看到的文件范围

在 writeup 中写到了访问这个链接可以查看到了 tips 这个是已经算好的了
http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=127.0.1.1%2F188f6594f694a3ca082f7530b5efc58dedf81b8d/admin/%20HTTP/1.1%0aHost:%20localhost%0aRange:%20bytes=370-%0a%0a

Range: bytes=370- 这个就是那个特殊的请求头的
这里有这个的参考资料 http://www.liqwei.com/network/protocol/2011/886.shtml
这里摘抄一部分内容

所谓断点续传，也就是要从文件已经下载的地方开始继续下载。在以前版本的 HTTP 协议是不支持断点的，HTTP/1.1 开始就支持了。一般断点下载时才用到 Range 和 Content-Range 实体头。

Range

用于请求头中，指定第一个字节的位置和最后一个字节的位置，一般格式：

Range:(unit=first byte pos)-[last byte pos] 后一个 last byte pos 是可以省略的

Content-Range

用于响应头，指定整个实体中的一部分的插入位置，他也指示了整个实体的长度。在服务器向客户返回一个部分响应，它必须描述响应覆盖的范围和整个实体长度。一般格式：

Content-Range: bytes (unit first byte pos) – [last byte pos]/[entity legth]

所以可以构造 Range:%20bytes=0- Range:%20bytes=10-….在 HTTP 头里一直递增上去查看 admin 内容

响应头中也有 Content-Range 的信息
Content-Range: bytes 10-429/430

然后 完整的 url

http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=127.0.1.1%2F188f6594f694a3ca082f7530b5efc58dedf81b8d/admin/%20HTTP/1.1%0aHost:%20localhost%0aRange:%20bytes=370-%0a%0a

在极端的情况下 如果只能看到一个字符，就可以做个字典用 burp 去 GET 爆破 然后把所有字符连接起来 这样子比较快 当然也可以简单的 get 网络编码

Ox05 HTTP_HOST
然后看到了提示

“HTTP_HOST”
当前请求的 Host: 头信息的内容。
这个是我们可控的
相关的资料可以参考 rfc2616 http://datatracker.ietf.org/doc/rfc2616/

最后还是和之前一样多次尝试获得 Range: bytes=80-
http://58.229.183.24/188f6594f694a3ca082f7530b5efc58dedf81b8d/index.php?url=127.0.1.1%2F188f6594f694a3ca082f7530b5efc58dedf81b8d/admin/%20HTTP/1.1%0aHost:%20hackme%0aRange:%20bytes=80-%0a%0a