SSRF攻击文档翻译

wing / 2018-07-01 17:06:52 / 浏览数 8695 安全技术 WEB安全 顶(0) 踩(0)

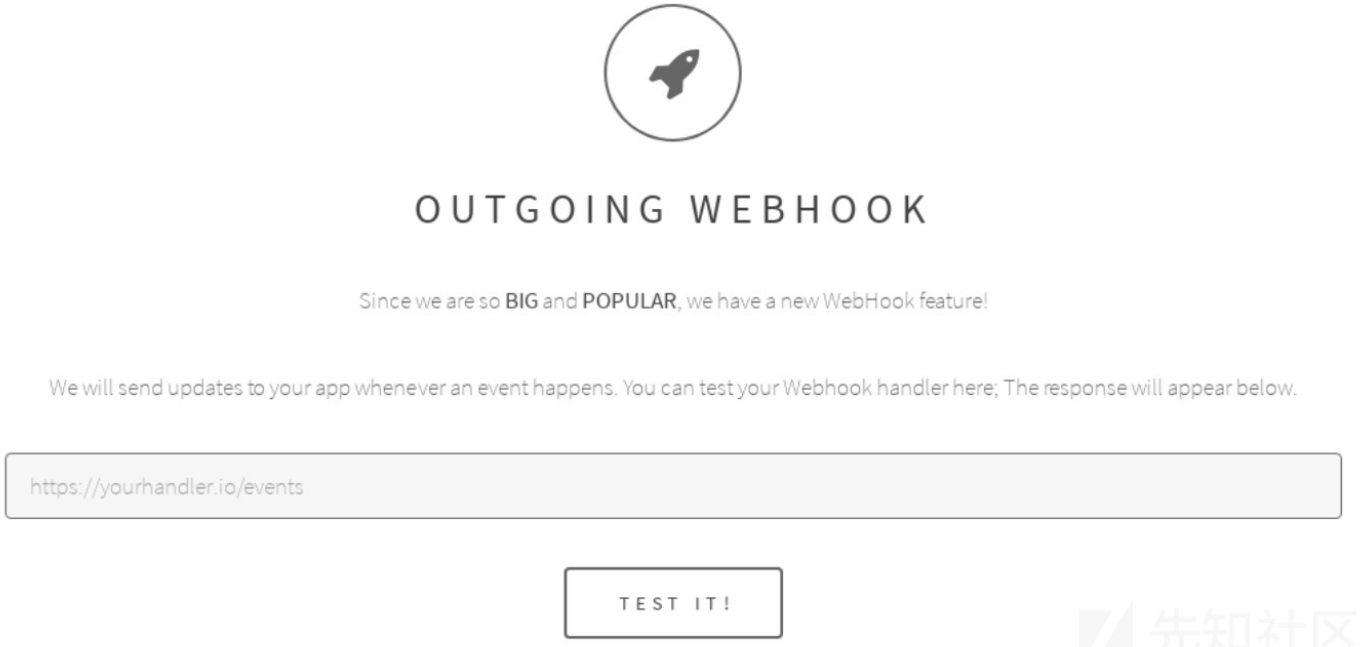
SSRF攻击文档翻译

Part1 基础知识

Hello world！
我将在本文向您介绍服务器端请求伪造(SSRF)的概念，它是客户端请求伪造（CSRF）的表兄弟，QWQ，在开始之前，我只是简单的介绍一下这个概念的基础知识，以便于
in Paris，Hackfest)和Orange
Tsai(DEFCON)，他们在很大程度上激发了本系列的内容，并以某种方式激发了我对这种技术的兴趣。我已经提供了一个SSRF模拟环境供您自己测试。文中所有的例子也是
那么服务器端请求伪造到底是什么？就是让服务器去请求你通常请求不到的东西，比如内网资产。这不是不可能，因为Web服务器通常可以访问比外部代理更多的资源，因
`in-depth`方法的局限性。
深度防御是一种信息保障（IA）概念，其中在整个信息技术（IT）系统中放置多层安全控制。
其目的是在安全控制失败或利用漏洞的情况下提供多余的保障，该漏洞可以涵盖系统生命周期期间的人员，程序，技术和物理安全方面。

SSRF攻击并不是一个新鲜的事物，但趋势正在增多并暴露原始攻击面。从现在来说，我们
将假设有一个全新的Web应用程序，它已经变得非常流行，现在已经开始实现诸如`REST API`和定制`WebHooks`等整洁的功能。

当然，你希望你的用户能够在发布之前测试他们的WebHook处理程序。在这种情况下，你需要给他们一个很好的调试接口：



因此，如果您的用户在`https://yourhandler.io/events`上监听`REST API`
，客户将收到测试事件并收到一些调试信息，例如来自其服务器的响应内容和状态代码。

如果我们处于攻击者的位置，我们现在有一台服务器愿意以我们的名义向任意位置发送HTTP请求，并向我们提供它得到的响应！问题是，它不能确保我们输入的URL实际上
Internet地址。如果不是事件处理程序，我们输入类似`http://127.0.0.1:8080`的内容怎么办？那么，我们现在在主机上有一个端口扫描器，超出了防火墙或安全组，并

那么，让我说一下WHOIS的一个请求，告诉您易受攻击的Web应用程序的公有IP地址是AWS基础架构的一部分，为什么不尝试去询问AWS元数据服务？

SEE THE RESULT!

我们现在知道我们Web应用程序的VPC地址空间。元数据服务可以提供非常有用的信息，但是如果我们不限制在Web上呢？让我们记住一个URL的结构：

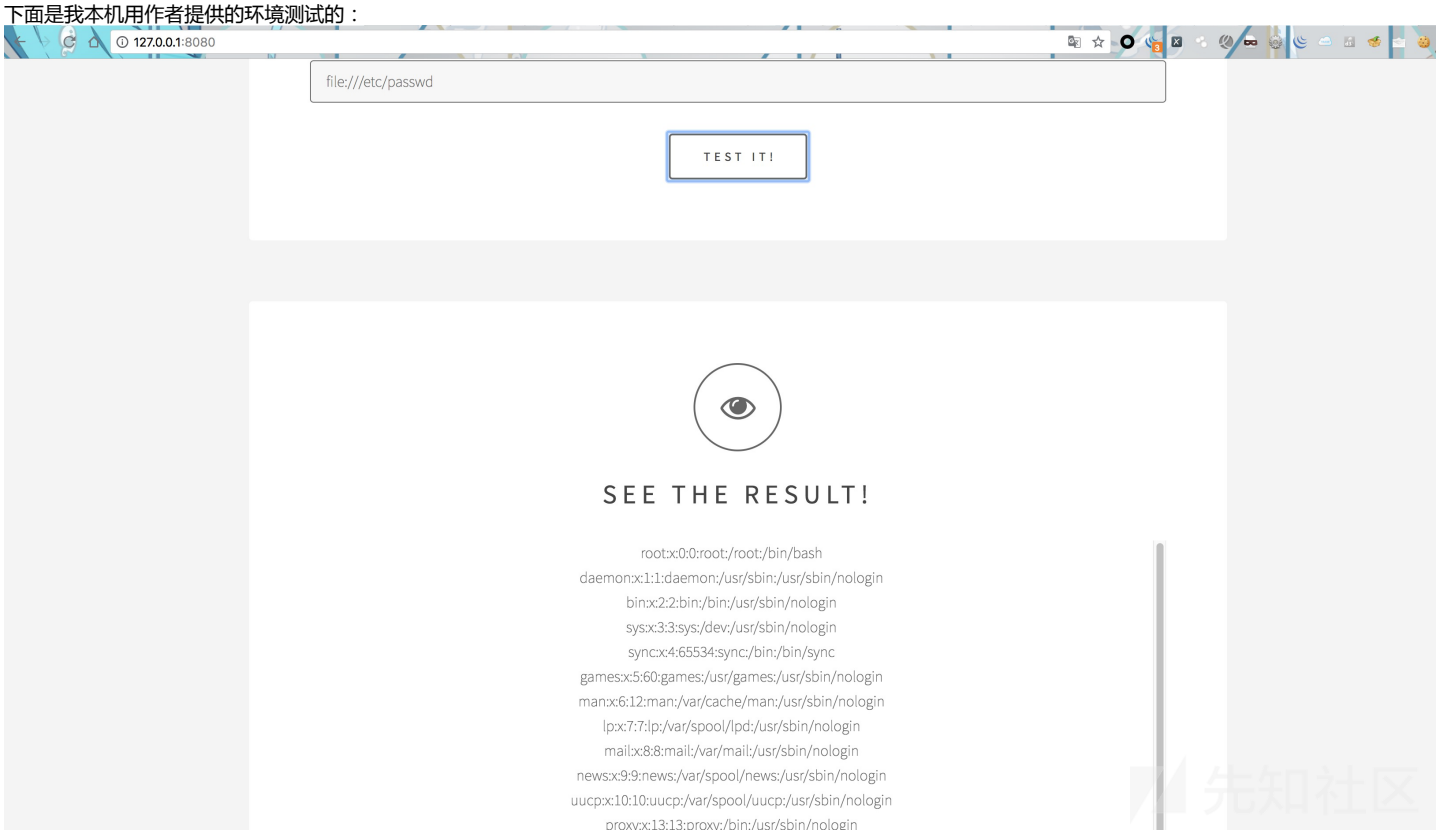
```
scheme://user:pass@host:port/path?query=value#fragment
```

可以尝试访问主机的文件系统。

SEE THE RESULT!

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin

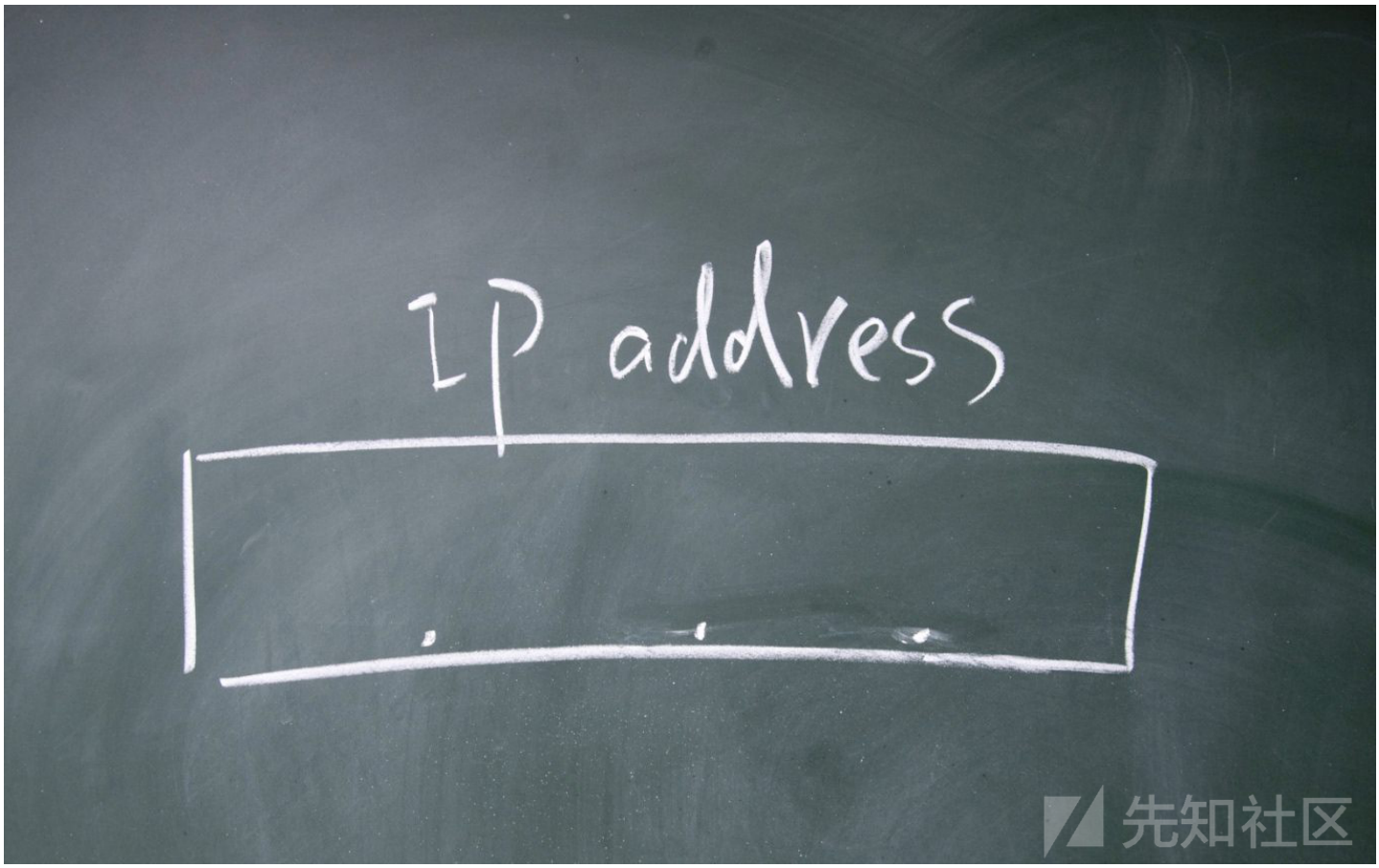
下面是我本机用作者提供的环境测试的：



YEAH！作为示例，我们现在能够挖掘配置文件和源代码文件，查找数据库凭据。

结论
如果您的应用程序向外部资源发出请求，请确保它们在所有情况下都是外部模式。我隐约听到你想说在这个例子的情况下，对用户输入进行简单的正则表达式过滤就可以实现
`white-listing`

Part2 Fun with IPv4 addresses



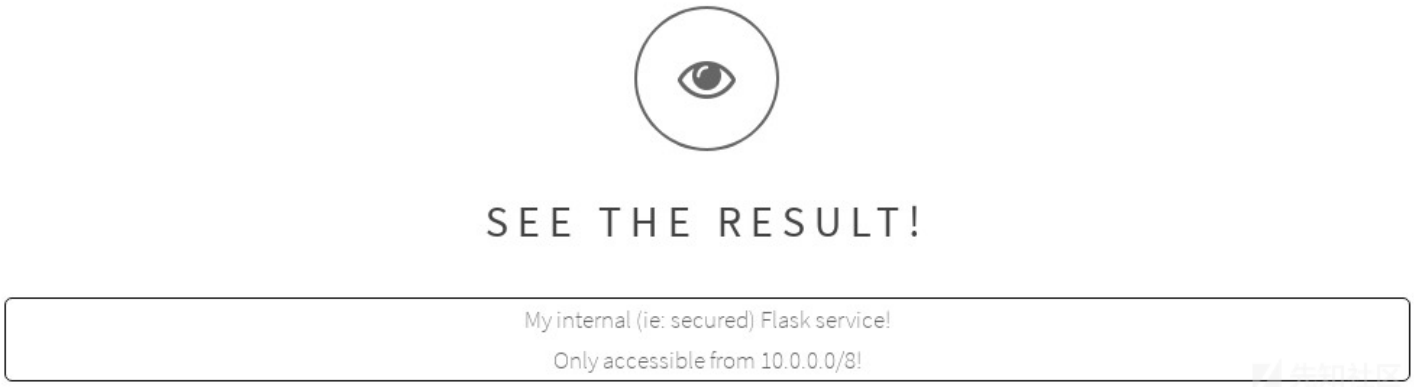
这是我的文章关于服务器端请求伪造攻击（SSRF）的第二部分,我必须赞扬NicolasGrégoire（Hack in Paris，Hackfest）和Orange Tsai（DEFCON），他们都强烈启发了这篇文章（第2部分特别广泛地使用了Nicolas的技术，我希望他把它作为贡品而不是抄袭，如果他看到这个）。

假设您已经意识到在新的WebHook功能中存在SSRF攻击的可能性，并决定以限制请求http://和禁止请求的方式保护10.0.0.3，让外部不应该看到内部的一些私有服务。您决

```
if (preg_match('#^https?://#i', $handler) !== 1) {
 echo "Wrong scheme! You can only use http or https!";
 die();
} else if(preg_match('#^https?://10.0.0.3#i', $handler) === 1) {
 echo "Restricted area!";
 die();
}
```

免责声明：这不是保护您的服务器的好方法，请不要这样做。还要注意WebHook只是使用一个简单的PHP curl函数，没有什么特别的。
scheme://留在第三部分讲，那怎么请求10.0.0.3?

该死的，怎么可能 - 我们在没有DNS的情况下以某种方式解决这个问题？让我们尝试一些不一样的，我会稍后解释; 让我们尝试一个请求：http://167722163



等等，那里发生了什么？请记住，IPv4地址只是网络流中相对于OSI第3层（IP）的四个字节。为方便起见，我们通常将其表示为四个数字，但此值的整数也完全有效。让我

```
■■■■■ 10.0.0.3
■■■■       00001010■00000000 00000000 00000011
■■■■■   0A.00.00.03
■■■       167772163
```

tips：php中有ip2long函数
http://aokunsang.iteye.com/blog/622498
现在，还有什么可以是我们能尝试的？肯定有很多方法可以表示32位;
其中最明显的是十六进制：http://0x0A000003。你猜怎么着？它工作了，它绕过了正则表达式过滤器，并且php-curl给力的解释为IPv4地址！
计算机世界中另一个鲜为人知的数字文字变体是八进制表示：在许多语言中，如果数字中最不重要的位置为零，则将其解释为base-8而不是我们人类友好的base-10小数点。

结论

从上文，我们可以尝试各种不同的方法;
我不知道如何在IP解析器中处理怎么处理的，但将十六进制中的.部分分开也是有效的：http://0x0A.0x00.0x00.0x03确实也是一个有效的请求http://012.00.00.
我无法在我的设置中重现此内容，但如果您有任何想法，请发表评论。

最后，我只想指出，在保护内部服务器时，正则表达式网络过滤并不是一种可行的方法。最好的方法是使用实□□际了解TCP /
IP逻辑的网络实用程序，以及网络协议创建者在实施这些标准时所考虑的所有细节。

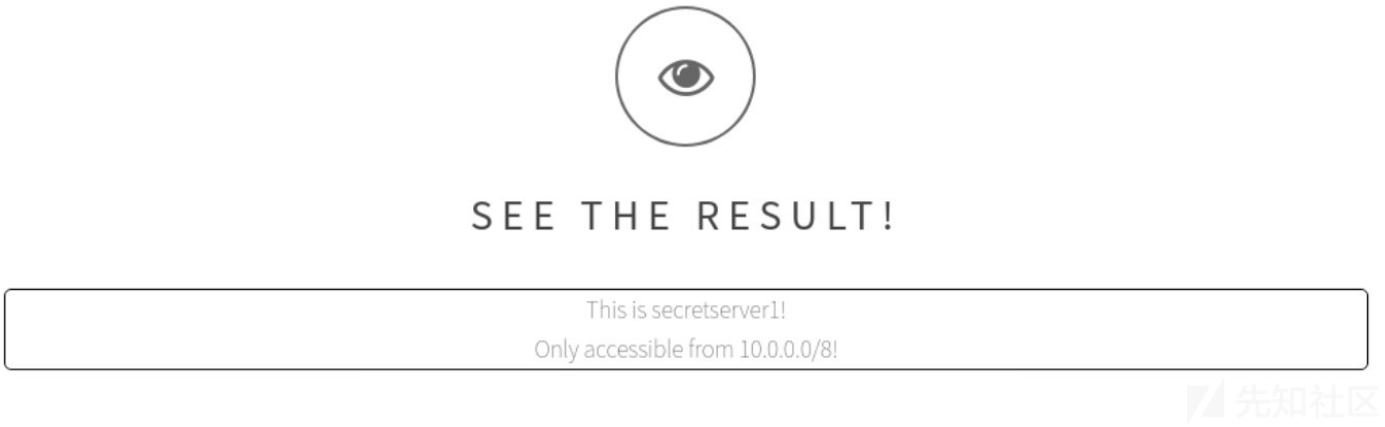
Part3 其他高级技术

这次我们的WebHook代码是用Python Flask编写的。我们还有其他secret.corp内部服务器位于10.0.0.3。请记住我对第2部分的结论，可以将其改写为：
■■■■■■■■■■; ■■■■■■■■■
所以你完全可以做到这一点，并使用Python 2.7的hostname库，在提交的url部分实现了一个过滤器：

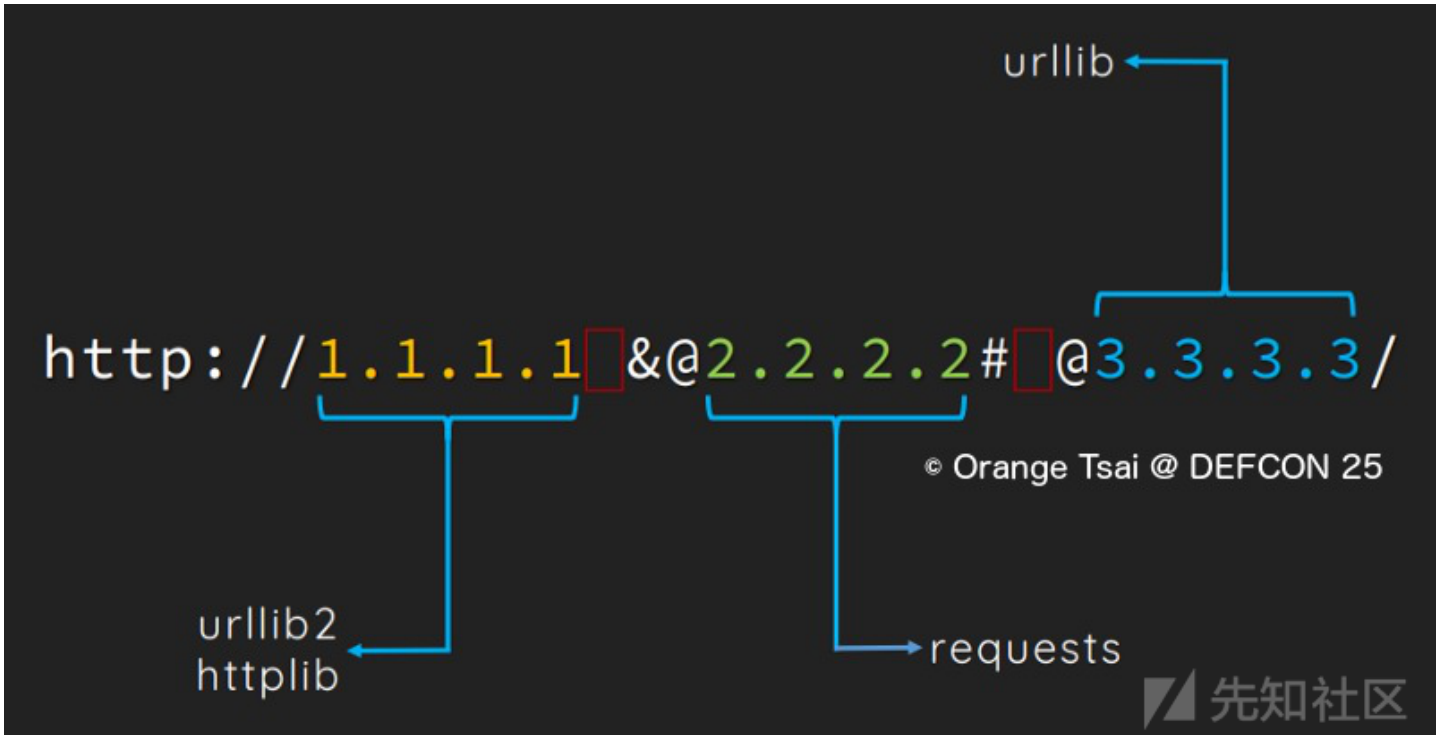
```
url = request.form ['handler']
host = urlparse.urlparse■url■.hostname
if host =='secret.corp'■return'Restricted
   Area■'
else■
   return urllib.urlopen■url■.read■■
```

让我们尝试通过尝试访问这个服务器`http://secret.corp`：

似乎被过滤了，现在，为了科学，我们来尝试一下`http://google.com# @secret.corp`（注意URL中间的空格字符）：



等等，又发生了什么事？我们使用标准解析库的过滤器被绕过并授权它,达到访问不应该被请求的资源？嗯，是的，不是;
这里的问题是，我们使用一个库来过滤hostname另一个库来执行实际的请求，而且它们不会以相同的方式在URL中间插空格！该urlparse库解析网址为google.com。而url
但情况变得更糟.....这是Orange Tsai演讲的一部分幻灯片：



正如你所看到的，Python库都有自己的方式来处理URL中的空格，如果你不在整个应用程序中坚持一种方式，那么在处理这些时可能会出现意想不到的结果。以下是官方RF

```
The authority component is preceded by a double slash (“//”) and is terminated by the next slash (“/”), question mark (“?”), o
```

权限组件前面加双斜杠，并以/，？，#结束，或者结束的URI。
从这个定义来看，库表现得恰到好处是模棱两可的; 我个人认为没有任何事情完全符合这种行为。

未来的工作：协议注入

作为结论，我想介绍协议注入的概念：请记住第1部分中的URL结构：

```
scheme://user:pass@host:port/path?query=value#fragment
```

怎么可能绕过一个scheme://只能访问HTTP（S）协议的固定版本？符合换行注入技术。现在，在写这篇文章的时候，我还没有在我的SSRF实验中将它作为一个Web服务器
Postfix安装。我们的WebHook服务器只能发出HTTP请求。但是，如果我们能够\r\n像在主机名中一样注入换行符（），并且库接受它，则可以在SSL握手期间将有效的命令

```
Request:
url = https://mail.corp\r\nHELO web.corp\r\nMAIL FROM...:25/
Response:
SMTP: 502 5.5.2 Error: command not recognized <SSL Gibberish>
SMTP: 250 web.corp
SMTP: 250 2.1.0 Ok
...
SMTP: 502 5.5.2 Error: command not recognized <SSL Gibberish>
```

这里的技巧是，在SSL握手期间，主机名被完全使用，如果它们被请求的库接受，则包括它的换行符。问题是，SMTP将换行符解释为“命令结束”信号，因此我们可以在SSL握
，但我的第一个Wireshark捕获的包是有用的。请注意，同样的技巧应该适用于任何基于文本的协议：例如，我可以使用它来注册到内部SIP服务器。如果您对我有实施建议
Tsai和NicolasGrégoire的工作！

https://github.com/m6a-UdS/ssrf-lab

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板